Proceedings of ICCM 2020

18th International Conference on Cognitive Modelling[1]

Edited by

Terrence C. Stewart

1 Co-located with the 53rd Annual Meeting of the Society for Mathematical Psychology and held online

Preface

The International Conference on Cognitive Modeling (ICCM) is the premier conference for research on computational models and computation-based theories of human cognition. ICCM is a forum for presenting and discussing the complete spectrum of cognitive modelling approaches, including connectionism, symbolic modeling, dynamical systems, Bayesian modeling, and cognitive architectures. Research topics can range from low-level perception to high-level reasoning. In 2020, ICCM was jointly held with MathPsych – the annual meeting of the Society for Mathematical Psychology. While ICCM and MathPsych were originally to be held in Toronto, Canada from July 25th-28th, the ongoing COVID-19 pandemic made this impossible. Instead, the conference was held online from July 20th to July 31st, using a combination of pre-recorded videos, live discussions, and custom software developed by the Society for Mathematical Psychology.

Acknowledgements

We would like to thank the Society for Mathematical Psychology (SMP) for their tremendous work in converting this conference to an online venue. In particular, the MathPsych Conference Chair (Joachim Vandekerckhove) guided the development of a custom website, and the officers of the SMP (especially Leslie Blaha) gave much-needed logistical support. ConfTool was used to manage submissions and reviews.

Papers in this volume may be cited as:

Lastname, A., Lastname, B., & Lastname, C. (2020). Title of the paper. In Stewart, T. C. (Ed.). *Proceedings of the 18th International Conference on Cognitive Modelling* (pp. 6-12). University Park, PA: Applied Cognitive Science Lab, Penn State.

ISBN-13: 978-0-9985082-4-5, published by the Applied Cognitive Science Lab, Penn State.

Conference Committees

General and Program Chair

Terrence C. Stewart	National Research Council of Canada

Program Committee

Leslie Blaha	Air Force Research Laboratory
Jelmer Borst	University of Groningen
Edward Cranford	Carnegie Mellon University
Aji Ghose	Birkbeck, University of London
Olivia Guest	RISE
Theodros Haile	University of Washington
Alex Kelly	Pennsylvania State University
Wouter Kruijne	University of Groningen
Othalia Larue	Wright State University
Christian Lebiere	Carnegie Mellon University
Peter Lindes	University of Michigan
Konstantinos Mitsopoulos	Carnegie Mellon University
Jacob David Oury	Pennsylvania State University
David Peebles	University of Huddersfield
Kai Preuss	TU Berlin
Roussel Rahman	Rensselaer Polytechnic Institute
Sarah Ricupero	Pennsylvania State University
Frank Ritter	Pennsylvania State University
Nele Russwinkel	TU Berlin
Dario Salvucci	Drexel University
David Marc Schwartz	Pennsylvania State University
Catherine Sibert	University of Washington
Sterling Somers	Carnegie Mellon University
Bryan Stearns	University of Michigan
Andrea Stocco	University of Washington
Niels Taatgen	University of Groningen
James Chapman Treyens	University of Washington
Robert L. West	Carleton University
Yuxue Yang	University of Washington

Table of Contents

Joke Recommender System Using Humor Theory

Soumya Agrawal (agraw105@purdue.edu)
Julia Taylor Rayz (jtaylor1@purdue.edu)
Purdue University
West Lafayette, IN 47906 USA

Abstract

In this paper, we propose a methodology that aims to develop a recommendation system for jokes by analyzing its text. This exploratory study focuses mainly on the General Theory of Verbal Humor and implements the knowledge resources defined by it to annotate the jokes. These annotations contain the characteristics of the jokes and hence are used to determine how alike the jokes are. We use Lin's similarity metric and Word2vec to calculate the similarity between different jokes. The jokes are then clustered hierarchically based on their similarity values for the recommendation. Finally, for multiple users, we compare our joke recommendations to those obtained by the Eigenstate algorithm which does not consider the content of the joke in its recommendation.

Keywords: Computational humor; General theory of verbal humor; Clustering; Joke similarity

Introduction

Humor is an interesting phenomenon that can be identified most of the time but is very difficult to 'define' (McGhee & Pistolesi, 1979). Yet, its importance becomes more evident with humorless technological advances. Humor is much more than just a source of entertainment; it is an essential tool that aids communication. Various empirical findings have confirmed that stress and depressing thoughts can be regulated with the help of humor (Francis, Monahan, & Berger, 1999). Positive psychology, a field that examines what people do well, notes that humor can be used to reduce tension, make friends, make others feel good, or to help buffer stress (Lurie & Monahan, 2015) (Ruch & Heintz, 2016).

The need for humor in a computerized setup is often discussed and many researchers have presented their findings. Some of the applications of computational humor are human-computer interfaces (Morkes, Kernal, & Nass, 1998), education (McKay, 2002), edutainment (Stock, 1996), understanding how human brain works (Binsted et al., 2006; Ritchie, 2001), etc.

The advancements in AI have sowed the seeds of the idea that computers can understand the human language. Since humor is a ubiquitous aspect of the human experience, it is fair to expect the computers to take into consideration the humorous facet. Almost two decades ago, it was pointed out that if computer systems can incorporate humor mechanisms, then these systems would appear to be more user-friendly hence less alien and intimidating (Binsted, 1995). This statement still holds and to achieve this, one of the key things to consider is that different people find different things funny which makes research in this field both challenging and interesting.

Verbally expressed or verbal humor is a common form of humor, and one of the subclasses of verbal humor is the joke. A joke can be defined as "a short humorous piece of literature in which the funniness culminates in the final sentence" (Hetzron, 1991). This paper focuses on verbally expressed humor with the help of jokes.

The motivation for this research comes from the observation that the smart assistants like Alexa and Siri recite the same jokes to all the users without considering their humor preferences. The idea behind this research is to come closer to understand human humor preferences and recommend jokes based on it. We propose a framework to recommend jokes to the users by taking into account the text of the joke as well as the liking of the users. Our assumption is that individuals like certain categories or types of jokes. These types can be identified through the individual's funniness ratings.

This framework is centered on the identification and quantification of similarity between jokes. The General Theory of Verbal Humor states that jokes can be represented and compared with the help of six knowledge resources (Attardo & Raskin, 1991). We use these knowledge recourses to find joke similarity in the Jester Dataset. Once similar jokes are identified, we explore whether subject ratings confirm the similarity.

There exists a joke recommendation system, Jester, (Goldberg, Roeder, Gupta, & Perkins, 2001) but it considers the users and the text of the joke as a black box and relies solely on the user ratings for the recommendation. It works as a baseline model to our proposed model and we compare the joke recommendations to the same user by both the models. We also analyze the ratings given by the users to the jokes that are considered similar to our model.

Humor Theories

Humor studies date back to the era of Plato (*Philebus*) and Aristotle (*Poetics*). There are three major classes of humor theory: superiority theories, release/relief theories, and incongruity theories. The general idea behind superiority theories was that people laugh at other people's misfortunes since it makes them feel superior to them (Attardo, 1994) (Raskin, 1985). Release/relief theories assert that humor and laughter are a result of the release of nervous energy (Meyer, 2000). The family of incongruity theories states that humor arises when something which was not anticipated happens (Raskin, 1985). There has been a debate among various

thinkers if incongruity alone can be considered to be sufficient enough to be able to mark something as funny (Suls, 1977).

This gave birth to the Incongruity-Resolution theories which focused not only incongruity but also on its realization and resolution. Suls (1972) proposed a two-stage model that stated that when there is some incongruity in the text, if one can resolve it then it's a joke otherwise the text leads to puzzlement and no laughter (Ritchie, 1999). Another model to resolve incongruity was summarized by Ritchie (1999) as the surprise disambiguation model which states that the setup of the joke has two different interpretations out of which one is more obvious than the other. The hidden meaning of the text is triggered once the punchline is reached.

Raskin's Script-based Semantic Theory of Humor (Raskin, 1985) is the first linguistic theory of humor. It is regarded as neutral concerning the three classes of humor theories. SSTH states that a joke carrying text should be fully or partially compatible with two scripts and these scripts must oppose. Raskin introduced several types of script oppositions, such as real/unreal, actual/non-actual, good/bad, life/death, sex/non-sex. The following joke is analyzed in Raskin (Raskin) with the scripts of Doctor and Lover[1] being the two scripts that overlap and oppose.

Joke$_1$: *'Is the doctor at home?' the patient asked in his bronchial whisper. 'No,' the doctor's young and pretty wife whispered in reply. 'Come right in.*' (Raskin, 1985)

The joke evokes the script of a Doctor due to the words "doctor", "patient" and "bronchial". The second script, Lover, is triggered by the words "no" as well as the description of the doctor's wife. The wife's reply is incongruous to the first script, and thus the second script emerges, which makes the punchline, "come right in" explainable. The joke is said to have a partial script overlap between Doctor and Lover – both scripts contain a person that comes to the doctor's house for a visit – and since these scripts are opposing each other based on sex/non-sex, the text is considered a joke (Attardo, 1994) (Raskin, 1985).

Attardo and Raskin (1991) revised the SSTH into General Theory of Verbal Humor which stated that the jokes can be described using six knowledge resources (KRs) which are ordered hierarchically: script overlap/opposition (SO), logical mechanism (LM), situation (SI), target (TA), narrative strategy (NS), and language (LA). Upon empirical verification of the KR hierarchy, LM was found to behave differently than predicted (Ruch, Attardo, & Raskin, 1993). GTVH also made the comparison of jokes possible with the KRs. The higher the number of common parameters in jokes, the higher is joke similarity. Additionally, jokes that differ only in SO are less similar than the jokes that differ only in LM, than the jokes that differ only in SI and so on. For example, the following jokes are introduced in Attardo & Raskin (1991) to illustrate the comparison:

Joke$_2$: *"How many Irishmen does it take to screw in a light bulb? Five. One to hold the light bulb and four to turn the table he's standing on."*

Joke$_3$: *"How many Poles does it take to wash a car? Two. One to hold the sponge and one to move the car back and forth".*

Joke$_4$: *"Do you think one Pole can screw in a light bulb?" "No." "Two?" "No." "Three?" "No. Five. One to screw in a light bulb and four to turn the table he's standing on."*

The KRs representing these jokes are represented in Table 1:

Table 1: Joke Comparison (Attardo & Raskin, 1991)

KR	Joke$_3$	Joke$_4$	Joke$_5$
SO	Dumbness	Dumbness	Dumbness
LM	Figure-Ground Reversal	Figure-Ground Reversal	Figure-Ground Reversal
SI	Light Bulb	Car Wash	Light Bulb
TA	Irish	Poles	Poles
NS	Riddle	Riddle	Ques -Ans
LA	LA 1	LA 1	LA2

Here, jokes 3 and 4 differ in three of the parameters, namely, LA, NS, and SI; jokes 2 and 3 differ in two of them, namely TA and SI; and jokes 2 and 4 in three of them, namely LA, NS and TA. Jokes 2 and 3 are the most similar since they differ in only two knowledge resources. Since SI is placed at a higher level in the hierarchy, jokes 3 and 4 are the least similar even though they have the same number of different KRs as jokes 2 and 4. This paper will rely on this theory to process humor computationally.

Methodology

We assume that previously unseen jokes should be recommended to users as well as jokes that have been rated by others (and thus, have been seen by the system). This means that the content of the jokes, not just the user ratings, has to be taken into consideration. To do so, we develop a methodology to compare jokes based on their content, find their similarity, and then cluster them accordingly. The jokes which are clustered together -- and have at least one highly rated joke – serve as the recommendations for the users.

Corpus

This paper adopts jokes from the Jester dataset. We use version 3[2] of the dataset which is an updated dataset of the previous versions. Version 1 has rating values from -10 to +10 of 100 jokes collected between April 1999 to May 2003 and the version 2 has 50 more jokes with 115,000 new ratings collected between November 2006 to May 2009. Overall, the version 3 of the dataset has over 1.8 million continuous

[1] The naming of the scripts has been debated in various humor papers. The Ontological Semantic Theory of Humor (Raskin, Hempelmann, & Taylor, 2009) can be used to identify the scripts without committing to their naming.

[2] http://eigentaste.berkeley.edu/dataset/

ratings of 150 jokes from 54,905 anonymous users which were collected from November 2006 to March 2015. It should be noted that many jokes in the dataset are no longer relevant (out of date), but they can nevertheless be used to test the methodology. The dataset consists of a set of 8 jokes termed as *gauge set,* as these jokes are rated by all the users. The remaining non-gauge jokes have a very sparse rating matrix since around 82% of the user ratings are null.

All jokes from the dataset have been annotated with the six knowledge resources as defined by GTVH by the domain knowledge experts. We wish to point out that two pairs of jokes in the dataset are identical and we decide to remove the duplicate from measuring joke similarity.

Baseline Model

The joke recommendation system (Goldberg et al., 2001) is based on a constant-time collaborative filtering algorithm that recommends jokes to the users based on their rating of the gauge set jokes. To overcome the problem of the sparse rating matrix, the model is built on the ratings of gauge set jokes only. The algorithm uses Principal Component Analysis (Pearson, 1901) to optimally reduce the dimension of the data to two. Since the projected data had a high concentration around the origin, a clustering algorithm was developed which recursively bisected the data near the origin into rectangle-shaped clusters. Whenever a user enters the system the ratings of the gauge set are collected which helps the algorithm to determine which cluster to place the user in. For each cluster, the mean of the non-gauge jokes ratings is calculated which are sorted in the decreasing order and this yields a lookup table. The lookup table is referenced every time a joke is recommended to the user.

GTVH-based Framework for Joke Similarity

We analyze the text of the jokes based on the GTVH knowledge resource (KR) annotations done by the domain knowledge experts. We focused on SO, LM, SI and TA, as LA value should differ for every joke and most jokes in the dataset have the same NS value. To find the pairwise similarity of the jokes we compare the instances of the corresponding KRs. Attardo and Raskin (1991) do not define the hierarchy of each of the KRs, however, a sketch of SO hierarchy can be reconstructed from Raskin (Raskin), and a partial hierarchy of LMs can be found in (Attardo, Hempelmann, & Di Maio, 2002). We extended the hierarchies of SOs, LMs, and SIs based on the information from the jokes, using the methodology for ontology construction from the Ontological Semantic Technology – a foundation of the Ontological Semantic Theory of Humor.

To construct a hierarchy, each of the entities are described by their properties. The properties and their values serve the guiding principle for hierarchy construction (Taylor & Raskin). Each of the children differ from the parent by a property, and the siblings should differ from each other only by the values of the chosen property. Once the hierarchy is constructed, all descendants that do not have siblings are collapsed into a single node. In other words, no non-leaf node can have less than two children.

Joke Similarity

The joke similarity metric for each of the resources is motivated by Resnik (1995) model, that proposed to estimate the common amount of information by the information content of the least common subsumer of the two nodes. Lin (1998) extended this concept by adding that the similarity metric must also take into account the differences between the two entities. To compare each instance of SO, LM and SI, the following function is used:

$$\text{Similarity}\,(kr_a, kr_b) = \begin{cases} 1 & if\ kr_a = kr_b \\ 0 & if kr_a\ or kr_b is\ null \\ sim_{Lin}\,(kr_a, kr_b) & all\ other\ cases \end{cases}$$

where kr_a and kr_b are the instances of the same KRs and sim_{Lin} is Lin's similarity measure (Lin, 1998), adapted from Jurafsky and Martin (2018) used for word similarity:

$$sim_{Lin}\,(c_1, c_2) = \frac{2 * logP(LCS(c_1, c_2))}{logP(c_1) + logP(c_2)}$$

where P(c) is defined by as the probability that a random word selected in a corpus is an instance of concept c and LCS(c_1, c_2) is the lowest node in the hierarchy that subsumes both c_1 and c_2. In our case, c_1 and c_2 are instances of a hierarchy of SO, LM, or SI.

To compare TA instances, we use word embeddings. Recent advancements in NLP research has seen the popularity of word embedding models which represent the words as vectors in a predefined vector space. One such word embedding model is word2vec (Mikolov, Chen, Corrado, & Dean, 2013) which is a shallow neural network that takes a text corpus as input and returns the vector representations of the words. To compare TAs, we used a pre-trained googlenews model which has word vectors for 3 million words, obtained by training on a google news dataset of around 100 billion words. There were some TA annotations in our dataset that were not present in the word2vec-based model. To overcome this problem, we made appropriate replacements of those annotations, ensuring that the new annotations preserve the context.

$Joke_5$ and $Joke_6$ illustrate joke annotation and calculation of joke similarity. We provide a modified version of the $Joke_5$ in this paper due to a potentially offensive nature of the original and replace $Joke_6$ with a very close joke taken from another source:

$Joke_5$: *A guys walks into a bar and tells the bartender that he has the best Polish joke. "I am Polish," responds the bartender. "Don't worry, I will tell it slowly."*

$Joke_6$: *"What did the liberal arts major say to the engineering grad?" "Do you want fries with that?"*[3]

[3] https://upjoke.com/liberal-art-jokes

The GTVH-based annotations for $Joke_5$ and $Joke_6$ are shown below:

$$\begin{matrix} \textbf{SO} \\ \textbf{LM} \\ \textbf{SI} \\ \textbf{TA} \end{matrix} \begin{Bmatrix} \textbf{actual/non-actual} \\ \textbf{faulty reasoning} \\ \textbf{going to bar} \\ \textbf{poles} \end{Bmatrix} \begin{Bmatrix} \textbf{actual/non-actual} \\ \textbf{faulty reasoning} \\ \textbf{intellectual discussion} \\ \textbf{graduates}[4] \end{Bmatrix}$$

Since the instances of SO and LM are the same for both jokes, their corresponding similarity is 1. For TA, the word2vec similarity between *poles* and *graduates* is 0.046 using the methods defined by the Gensim library (version 3.8.1) on the pre-trained model. For SI, we look at the fragment of the SI hierarchy along with the P(c), as depicted in Figure 1. The nodes of interest are highlighted. This results in the following:

$$sim_{Lin}\left(SI_{going\ to\ a\ bar}, SI_{intellectual\ discussion}\right) = \frac{2 * \log(1)}{\log(0.0066) + \log(0.0066)} = 0$$

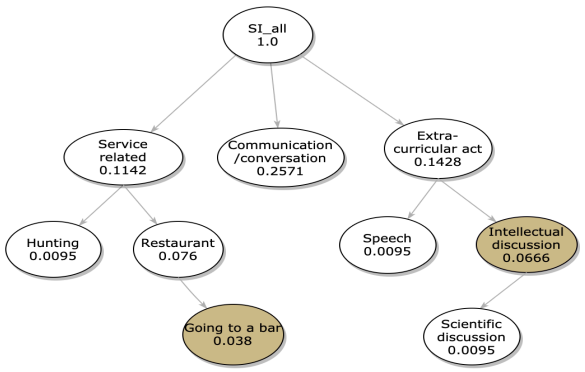


Figure 1: SI hierarchy fragment

To take into consideration the hierarchy of KRs themselves, as proposed by SSTH, we assign a weight, w_{SO}, w_{LM}, w_{SI}, and w_{TA}, to each of the KRs such that $w_{SO} < w_{LM} < w_{SI} < w_{TA}$:

$$\text{sim}(\text{joke}_i, \text{joke}_j) = \frac{[w_{so} \quad w_{LM} \quad w_{SI} \quad w_{TA}] \begin{bmatrix} \text{sim}(SO_{jokei}, SO_{jokej}) \\ \text{sim}(LM_{jokei}, LM_{jokej}) \\ \text{sim}(SI_{jokei}, SI_{jokej}) \\ \text{sim}(TA_{jokei}, TA_{jokej}) \end{bmatrix}}{w_{so}+w_{LM}+w_{SI}+w_{TA}}$$

For this paper, the following values are assigned: $w_{SO}=5$, $w_{LM}=4$, $w_{SI}=3$ and $w_{TA}=2$.

$$\textbf{sim}(\textbf{joke}_5, \textbf{joke}_6) = \frac{[\mathbf{5\ 4\ 3\ 2}] \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{0.046} \end{bmatrix}}{5+4+3+2} = \frac{\mathbf{0.092}}{\mathbf{15}} = \mathbf{0.649}$$

The value calculated after the weighted average is 0.649 which quantifies how similar $Joke_5$ and $Joke_6$ are.

[4] Annotation has been changed from *liberal arts graduate* to *graduates* since the former was not in word2vec-based-model

Joke Clustering

We aim to cluster the jokes so that the most similar ones are close to each other. To achieve this, we implement the hierarchical clustering algorithm in which we use the joke similarity values for the distance calculation. Figure 2 shows the dendrogram of the jokes after clustering.

Joke Recommendations

To recommend jokes to a user, we would identify the user's favorite joke with the help of the user ratings of the corpus. The joke which is the immediate sibling of the favorite joke in the dendrogram is recommended first. To recommend more jokes, we move up in the hierarchy and if there are multiple jokes available on the same level of the hierarchy, then a random selection of the jokes is done for that level.

Results

A qualitative evaluation was performed on the GTVH-based model. A user was randomly selected for comparison of the recommendations made by the baseline and the GTVH-based model. To compute the top recommend jokes from the GTVH-based model, we use the selected user's top-rated joke from the dataset which is known to our system as the favorite joke. The same user's ratings of the recommended jokes from both the models were used to compare them. Table 2 shows the results for five randomly selected users.

We are restricted in selecting the users due to the sparsity of the rating dataset which sheds light on one of the difficulties with working with this dataset. We meticulously select report results on the users who have rated the jokes in both the baseline and the GTVH, to ensure that the comparison of the two models is possible. The results for randomly selected 5 users are shown in Table 2. The highest-rated joke for user 1, as well as recommended jokes by the baseline and the GTVH-based model are presented as well. For user 1 in Table 2, we observe that the top joke recommended by the GTVH-based model (Joke 87) has a better rating than the top joke recommended by the baseline model (Joke 89). We can see by the text of the jokes that the favorite joke of user 1 and Joke 87 are very similar whereas Joke 89 is very different from these jokes. We provide the modified versions of some of jokes from the dataset for the analysis.

User1's favorite joke: *An artist has been displaying his paintings at an art gallery and he asked the owner if there had been any interest in his paintings. "I've got good news and bad news," says the owner. "The good news is that a gentleman inquired about your work and wondered if it would be worth more after your death. When I told him it would, he bought all ten of your paintings." "That's wonderful!" the artist says. "What's the bad news?" With concern, the gallery owner replied: "The man was your doctor."*

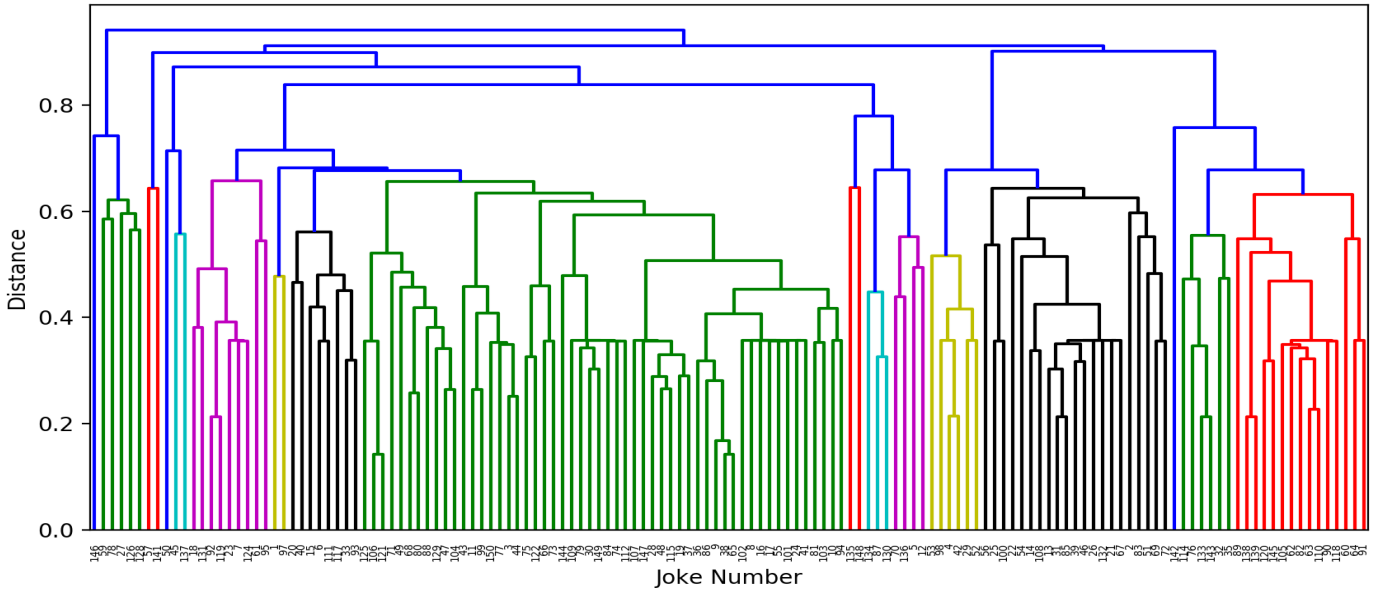


Figure 2: Hierarchical clustering of jokes

Table 2: Comparison of Recommended Jokes

	Baseline Model		GTVH-based Model	
	Top Recommended Joke	**Rating**	**Top Recommended Joke**	**Rating**
User 1	Joke 89	8.18	Joke 87	9.37
User 2	Joke 73	-1	Joke 42	5.71
User 3	Joke 53	3.56	Joke 72	3.46
User 4	Joke 5	9.87	Joke 112	0.93
User 5	Joke 89	4.56	Joke 126	5.62

Table 3: Cluster Analysis for the five selected users

	Cluster 1		**Cluster 2**		**Cluster 3**		**Cluster 4**		**Cluster 5**		**Cluster 6**		**Cluster 7**	
Joke Id	**92**	**119**	**106**	**121**	**38**	**65**	**31**	**85**	**133**	**143**	**138**	**139**	**110**	**63**
User 6	3.02	2.88	3.39	2.91	3.10	3.58	4.29	3.22	2.66	3.46	2.86	3.05	3.13	3.64
User 7	*2.45*	*4.22*	3.45	4.66	*4.91*	*3.13*	*1.02*	*4.14*	0.88	0.87	4.68	4.14	0.99	0.95
User 8	*1.40*	*4.31*	4.53	3.30	4.25	3.36	3.65	3.45	2.94	4.17	3.16	3.29	3.83	4.22
User 9	3.63	4.12	4.80	4.96	*4.79*	*1.35*	2.68	1.75	4.28	3.13	3.13	3.19	2.82	2.77
User 10	*0.19*	*4.31*	3.85	4.34	*3.90*	*1.21*	*4.31*	*0.13*	4.41	3.81	1.64	1.07	1.85	2.90

Joke 87: *A man after undergoing a routine physical examination receives a phone call from his doctor. The doctor says, "I have some good news and some bad news." The man says, "I want to hear the good news first." The doctor says, "The good news is, you have 24 hours to live." The man replies, "If this is good news then what's the bad news?" The doctor says, "The bad news is, I forgot to call you yesterday."*

Joke 89*: A radio conversation between a US naval ship and Canadian authorities... Americans: Please divert your course 15 degrees to the North to avoid a collision. Canadians: Recommend you divert YOUR course 15 degrees to the South to avoid a collision. Americans: You divert YOUR course. Canadians: No. You divert YOUR course. Americans: This is the second largest ship in the United States; Atlantic Fleet. We are accompanied by three destroyers, three cruisers and numerous support vessels. I demand that you change your course…., or countermeasures will be undertaken to ensure the safety of this ship. Canadians: This is a lighthouse. Your call.*

The proposed model works better than the baseline for users 1, 2 and 5, works moderately well for user 3, and fails to perform better for users 4. It should be noted that it is equally possible to find similar jokes to all highly rated jokes for a particular user. However, based on the results of user 4, we wanted to check whether highly similar jokes are typically rated similarly.

To further investigate how users rate jokes that are considered similar by the proposed model, we selected 5 users who have rated 140 jokes which the maximum number of jokes rated by any user. Also, we normalize the ratings to 0-5 for the experiment. We selected all the joke clusters which are formed near the distance value of 0.2 for the analysis. Table 3 lists 7 such clusters each consisting of 2 jokes for the comparison of user ratings of closely clustered,

and thus similar, jokes. We observe that the intra-cluster ratings of the users 6, 8 and 9 are largely similar for all the clusters where they differ greatly for user 7 and user 10. Both these users rate jokes in clusters 1, 3 and 4 were differently which implies that the jokes which are considered similar by the model are not equally appreciated by both the users. There are several explanations for this result, assuming that the ratings in the dataset accurately represent user preferences. The first one is that the similarity metric that we produced does not accurately represent joke similarity, and Target may need to be weighted heavier than the rest for the recommendation system. The second one is that the annotation of one of the jokes in the clusters may be flawed. The third, and perhaps most interesting one, is whether users tend to rate familiar jokes lower. We do not have the data on the ordering of jokes that were presented to the users, and thus, this is impossible to test this hypothesis. However, we can look at rating of almost identical jokes for these users.

As stated earlier, the corpus has ratings of two pairs of identical jokes, ratings of which for the same 5 users are summarized in table 3.

We can observe that users 7, 8 and 9 have given different ratings to identical jokes. Since the users were given a scroll button to rate the jokes, some variation in the ratings is acceptable but this difference is very high for users 7 and 8. It is tempting to conclude that the effect of a previously heard or rated joke must be taken into consideration while recommendations are made. It is also possible that for some users the almost identical jokes were presented very close to each other, while for others they were spread much farther apart among the 140 jokes. Lastly, the dataset also does not consider the effect fatigue effects of the users which may affect the ratings.

Table 3: Ratings given to Identical Jokes

	Identical Pair 1		**Identical Pair 2**	
User 6	5.12	0.62	3.15	1.37
User 7	*-5.25*	*4.68*	-7.31	-5.84
User 8	5.81	0.62	*9.62*	*1.68*
User 9	5.12	5.59	*3.53*	*7.28*
User 10	4.78	0.53	1.15	5.34

Conclusion

By taking into account the text of the jokes along with the user rating for joke recommendations, we observe that the model can select similar jokes, however, it is not clear that this by itself is the winning mechanism. To attain a more generalized framework for joke recommendations we need to 1) Conduct more research focusing on the manipulation of the weights assigned to the KRs 2) Collect user ratings while keeping track of the order of jokes in which they appear, thus taking into consideration the effect of a previously heard joke. We suspect that understanding user preference will go a long way towards more friendly interaction between various devices that have a functionality of telling a user a joke.

Acknowledgement

The authors are grateful to anonymous reviewers for their helpful comments and to Ken Goldberg for releasing the dataset to the public.

References

Attardo, S. (1994). Linguistic theories of humor (Vol. 1). *Berlin: De Gruyter Mouton. Barsoux, JL (1996). Why organisations need humour. European Management Journal, 14*(5).

Attardo, S., Hempelmann, C. F., & Di Maio, S. (2002). Script oppositions and logical mechanisms: Modeling incongruities and their resolutions. *Humor - International Journal of Humor Research, 15*(1). doi:10.1515/humr.2002.004

Attardo, S., & Raskin, V. (1991). Script theory revis (it) ed: Joke similarity and joke representation model. *Humor-International Journal of Humor Research, 4*.

Binsted, K. (1995). *Using humour to make natural language interfaces more friendly.* Paper presented at the Proceedings of the AI, ALife and Entertainment Workshop, Intern. Joint Conf. on Artificial Intelligence.

Binsted, K., Nijholt, A., Stock, O., Strapparava, C., Ritchie, G., Manurung, R., . . . O'Mara, D. (2006). Computational humor. *IEEE Intelligent Systems, 21*(2).

Francis, L., Monahan, K., & Berger, C. (1999). A laughing matter? The uses of humor in medical interactions. *Motivation and emotion, 23*.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *information retrieval, 4*.

Hetzron, R. (1991). On the structure of punchlines. *HUMOR: International Journal of Humor Research*.

Jurafsky, D., & Martin, J. H. (2018). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. *Inc. Upper Saddle River, NJ*.

Lin, D. (1998). *An information-theoretic definition of similarity.* Paper presented at the Icml.

Lurie, A., & Monahan, K. (2015). Humor, aging, and life review: Survival through the use of humor. *Social Work in Mental Health, 13*.

McGhee, P. E., & Pistolesi, E. (1979). *Humor: Its origin and development*: WH Freeman San Francisco.

McKay, J. (2002). Generation of idiom-based witticisms to aid second language learning. *Stock et al*.

Meyer, J. C. (2000). Humor as a double-edged sword: Four functions of humor in communication. *Communication theory, 10*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv.org*.

Morkes, J., Kernal, H. K., & Nass, C. (1998). *Humor in task-oriented computer-mediated communication and human-computer interaction.* Paper presented at the CHI 98

Conference Summary on Human Factors in Computing Systems.

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2*. doi:10.1080/14786440109462720

Raskin, V. (1985). *Semantic mechanisms of humor*. Dordrecht ; Boston: D. Reidel Pub. Co.

Raskin, V., Hempelmann, C. F., & Taylor, J. M. (2009). How to understand and assess a theory: The evolution of the SSTH into the GTVH and now into the OSTH. *Journal of Literary Theory, 3*.

Resnik, P. (1995). Using Information Content to Evaluate Semantic Similarity in a Taxonomy. *arXiv.org*.

Ritchie, G. (1999). *Developing the incongruity-resolution theory*.

Ritchie, G. (2001). Current directions in computational humour. *Artificial Intelligence Review, 16*.

Ruch, W., Attardo, S., & Raskin, V. (1993). Toward an empirical verification of the general theory of verbal humor. *Humor, 6*.

Ruch, W., & Heintz, S. (2016). The virtue gap in humor: Exploring benevolent and corrective humor. *Translational Issues in Psychological Science, 2*.

Stock, O. (1996). *Password Swordsfish: Verbal Humor in the Interface*: Citeseer.

Suls, J. M. (1972). A two-stage model for the appreciation of jokes and cartoons: An information-processing analysis. *The psychology of humor: Theoretical perspectives and empirical issues, 1*.

Suls, J. M. (1977). Cognitive and disparagement theories of humor: A theoretical and empirical synthesis. *Chapman and Foot, It's A Funny Thing Humour*.

Taylor, J. M., & Raskin, V. (2015). Comparing Apples and Orange Cottages: Classifications and Properties. *The Seventh International Conference on Advanced Cognitive Technologies and Applications*.

Decision Making as a Closed-Loop Process

Sophie-Anne Baker (sophie.baker@bristol.ac.uk)
Department of Engineering Mathematics, University of Bristol
Merchant Venturers Building, Woodland Road
Bristol, BS8 1UB, England, UK

Thom Griffith (thom.Griffith@bristol.ac.uk)
Department of Engineering Mathematics, University of Bristol
Merchant Venturers Building, Woodland Road
Bristol, BS8 1UB, England, UK

Giovanni Pezzulo (giovanni.pezzulo@istc.cnr.it)
Institute of Cognitive Sciences and Technologies
National Research Council, Rome, Italy

Nathan Lepora (n.lepora@bristol.ac.uk)
Department of Engineering Mathematics, University of Bristol
Merchant Venturers Building, Woodland Road
Bristol, BS8 1UB, England, UK

Keywords: Decision making; Embodied; Confidence; Closed-loop; Optimal decision making

Abstract

The theory of decision making has largely been developed as a disembodied open-loop process, however there is growing recognition that ecologically valid scenarios require integration of movement dynamics into current decision making theory, and a revision of what are considered to be core/fundamental decision components.

Here we develop the theory of decision making as a closed loop process, first exploring the role of confidence both as a neural computation within the loop, affecting movement dynamics and as a property of the egocentric frame with a causal influence on cognition. Secondly, we consider the relationship between closed-loop components/processing and stability — in embodied systems action is accumulated and so physical restrictions limit volatility, moreover the reciprocal relationship between movement and evidence processing means that this stabilisation may also happen on a neural level in the form of a biased gain during evidence accumulation, improving stability/convergence.

Finally, we examine closed-loop embodied decision making in the context of optimality — it is generally accepted that open-loop decision making is optimised to maximise reward via some form of Bayes' Risk, prescribing a speed-accuracy tradeoff in so doing. For closed-loop decision making however, the form of the 'objective function' is unknown, as such we consider higher level, ecologically inspired ideas of optimality such as adaptability to e.g. moving targets or nonstationarity, to explore this fundamental aspect of embodied decision making. Our results build on a growing body of work which points to embodiment as fundamental to understanding both behavioural and neural aspects of decision making.

Framework & Background

To explore the general principles of embodied decision making, we adopt the framework used by Lepora and Pezzulo (2015) based around a simple mouse-tracking 2-choice experiment. This framework separates distinctly the neural mechanisms from the behavioural – neural, in the form of evidence accumulation, and behavioural, in the form of spatial information; position and movement.

Under this framework, Lepora and Pezzulo (2015) find embodiment to have the key implication that a decision is not made simply when neural populations reach a threshold of activity, as has been recorded in immobilised decision making tasks in e.g. area LIP (Churchland, Kiani, & Shadlen, 2008), but when the action is complete, e.g. the cursor is placed on a target indicating the choice. To allow convergence to choice in a manner consistent with experimental data they consider two concepts – action preparation, and commitment – these bidirectionally connect the neural and behavioural components by utilising neurally represented evidence in movement modification (action preparation), and incorporating positional information into evidence accumulation (commitment), doing so renders the model entirely embodied and 'closes the loop'.

The strength of this model is its explanatory power using only evidence accumulation and spatial information. However, a number of questions remain unanswered; How does confidence affect action accumulation? Does an embodied closed-loop system have profound effects on behavioural and neural stability? Can we think of embodied closed-loop decision making in terms of optimality, as we do with traditional decision making paradigms? Within these broader questions are a number of consequential outcomes,

for example, what governs action initiation?

We develop the theory of decision making as a closed-loop process around these questions. With reference to experimental results, we build an intuition for the influence of decision components and the fundamental relationship between neural and behavioural mechanisms.

References

Churchland, A. K., Kiani, R., & Shadlen, M. N. (2008). Decision-making with multiple alternatives. *Nature neuroscience*, *11*(6), 693.

Lepora, N. F., & Pezzulo, G. (2015). Embodied choice: how action influences perceptual decision making. *PLoS computational biology*, *11*(4).

Degenerate Optimal Boundaries for Multiple-Alternative Decision Making

Sophie-Anne Baker (sophie.baker@bristol.ac.uk)
Department of Engineering Mathematics, University of Bristol
Merchant Venturers Building, Woodland Road
Bristol, BS8 1UB, England, UK

Thom Griffith (thom.Griffith@bristol.ac.uk)
Department of Engineering Mathematics, University of Bristol
Merchant Venturers Building, Woodland Road
Bristol, BS8 1UB, England, UK

Nathan Lepora (n.lepora@bristol.ac.uk)
Department of Engineering Mathematics, University of Bristol
Merchant Venturers Building, Woodland Road
Bristol, BS8 1UB, England, UK

Keywords: Decision making; Multiple-choice; Urgency signal; Decision threshold; Evidence accumulation

Abstract

Integration-to-threshold models of two-choice perceptual decision making have guided our understanding of the behaviour and neural processing of humans and animals for decades. Although such models seem to extend naturally to multiple-choice decision making, consensus on a normative framework has yet to emerge, and hence the implications of threshold characteristics for multiple choices have only been partially explored. Here we consider sequential Bayesian inference as the basis for a normative framework together with a conceptualisation of decision making as a particle diffusing in n-dimensions.

This framework implies highly choice-interdependent decision thresholds, where boundaries are a function of all choice-beliefs. We show that in general the optimal decision boundaries comprise a degenerate set of complex structures and speed-accuracy tradeoffs, contrary to current 2-choice results. Such boundaries support both stationary and collapsing thresholds as optimal strategies for decision-making, both of which result from stationary complex boundary representations.

This casts new light on the interpretation of urgency signals reported in neural recordings of decision making tasks, implying that they may originate from a more complex decision rule, and that the signal as a distinct phenomenon may be misleading as to the true mechanism. Our findings point towards a much-needed normative theory of multiple-choice decision making, provide a characterisation of optimal decision thresholds under this framework, and inform the debate between stationary and dynamic decision boundaries for optimal decision making.

Cognitive Mechanisms for Calibrating Trust and Reliance on Automation

Leslie M. Blaha (leslie.blaha@us.af.mil)
711^{th} Human Performance Wing, Air Force Research Laboratory, Pittsburgh, PA 15213 USA

Christian Lebiere (cl@cmu.edu)
Department of Psychology, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Corey K. Fallon (corey.fallon@pnnl.gov) and Brett A. Jefferson (brett.jefferson@pnnl.gov)
Pacific Northwest National Laboratory, Richland, WA 99352 USA

Abstract

Trust calibration for a human-autonomy team is the process by which a human adjusts their understanding of the automation's capabilities; trust calibration is needed to engender appropriate reliance on automation. Herein, we develop an Instance-based Learning ACT-R model of decisions to obtain and rely on an automated assistant for visual search in a UAV interface. We demonstrate that model matches well the human predictive power statistics measuring reliance calibration; we obtain from the model an internal estimate of automation reliability that mirrors human subjective ratings. Our model is a promising beginning toward a computational process model for trust and reliance for human-machine teaming.

Keywords: Cognitive architectures; Trust in automation; Human-machine teaming

Introduction

Trust calibration is the process team members go through to adjust their attitudes or expectancy of a favorable response from other teammates or of a positive outcome of a team effort (Lee & See, 2004). Research in both all-human teams and human-automation teams indicates that trust, and its behavioral proxy reliance, fluctuate over time. For human-automation teams, this calibration-related fluctuation reflects the human's process of learning when to rely on the automation. Fallon, Murphy, Zimmerman, and Mueller (2010) describe this as a sensemaking process wherein the human learns the conditions under which automation performs well, and how to properly interpret indicators provided by the machine system, to promote "appropriate use" (see also, Lee & See, 2004). Without calibration, the team may suffer from automation misuse or disuse by the human teammates (Lee & See, 2004; Parasuraman & Riley, 1997).

Lee and See (2004) formulated a conceptual model of appropriate trust formation. Within this model, trust calibration is part of a closed-loop process wherein people use task goals, context, and their own beliefs (including current trust level) to form an intent about using automation and then take Reliance Actions. The subsequent behavior of the automation and impact on the world (witnessed directly or via display) feed back into the human's Information Assimilation and Belief Formation processes, which then feed the Trust Evolution process, the Intent Formations, and Reliance Actions. Lee and See argued that information available about the automation and the results of Reliance Actions are critical to the trust formation process (as do Chen & Barnes, 2014; Fallon et al., 2010; Lyons et al., 2016, and many others). Merritt and Ilgen (2008) refer to trust emerging from interactions and experience with a system as history-based trust; they contrast history-based trust with other forms of trust, such as a person's general tendency to trust (dispositional trust; Jessup, Schneider, Alarcon, Ryan, & Capiola, 2019; Kramer, 1999; Merritt & Ilgen, 2008). It follows from this perspective that appropriate calibration can be defined as the degree of correspondence between a person's trust in automation and the automation's capabilities (see also, de Visser et al., 2020; Lee & Moray, 1994; Muir, 1987).

In this work, we develop a computational cognitive model of human decisions to rely on automation using Instance-based Learning Theory (IBLT; Gonzalez, Lerch, & Lebiere, 2003). Using an IBL model, we can explicitly model decisions about automation reliance and observe how those decisions are informed by task performance, transparency information, and the automation's behavior over time. In this way, we implement a computational processes mirroring elements of Lee and See's (2004) conceptual model.

Importantly, we do not explicitly incorporate a trust mechanism in the IBL model; rather, we maintain the perspective that trust is an attitude, separate from the cognitive decision making mechanisms, and that reliance is the behavioral indicator of trust. We seek to understand if and how trust calibration emerges from the task experience and decision making processes that are explicitly defined within the IBL model. In the remainder of this paper, we will describe an experiment on trust calibration measuring both intention formation and reliance action decisions. Then we will describe the IBL model and demonstrate its performance on this two-stage task. We will show that model reliance decisions mirror the human behavior, and we can extract an internal model bias that parallels human subjective judgments of automation reliability. We conclude that we have a strong candidate computational process model for trust calibration through experience with automation.

The Human-Automation Teaming Task

We leverage empirical data collected by Fallon, Blaha, Jefferson, and Franklin (2019) using the COgnitive Behavioral AnaLytics Testbed (COBALT). COBALT is an experimental interface developed by Fallon and Blaha to enable the study of trust, reliance on automation, task performance, and cognitive

Figure 1: A mid-trial screenshot of the COgnitive Behavioral AnaLytics Testbed (COBALT) task environment. This image shows an AUTOASSIST cue (pink arrow) selected for search, with an age + number text transparency cue (lower right). Readers are referred to Fallon et al. (2019) for more details about the interface.

workload while manipulating task characteristics, automation transparency, and interface design choices. COBALT is comprised of modular windows in which participants interact with automation to search for objectives in an aerial imagery. Figure 1 provides a screenshot of COBALT.

Two-stage Trial Structure Each trial of the task involves two stages: a decision stage and a search stage, as diagrammed in Figure 2. In the decision stage, participants must decide whether they would like the AUTOASSIST or COMMANDER to aid their visual search. Participants cannot perform the search without selecting an aid. Participants are provided transparency cues to help them decide if the AUTOASSIST will be a reliable choice.

The search stage begins as soon as the assist type is selected. Participants are tasked with searching for a predetermined objective randomly placed in the image. Participants are provided with search guidance in the form of an arrow overlaid on the search window. When reliable, this cue points directly to the objective; when unreliable, it points to some other random location. Participants can choose to follow the assist cue or search unguided.

Using the terminology of Lee and See's trust calibration model, the assist selection stage is an example of an automation reliance Intention Formation; participants indicate intention about using automation when they select the AUTOASSIST search aid. The search stage is an example of a Reliance Action. Participants following the AUTOASSIST cue are relying on automation; participants searching unguided are not.

Assist Types The assist types varied in their reliability and timing. The COMMANDER option provided a 100% reliable cue, always pointing to the objective. However, there was a 5 sec. delay between selecting the COMMANDER button and the COMMANDER assist arrow appearing on screen to aid the participant. While waiting for the COMMANDER assist, participants can search unaided for or wait for the COMMANDER cue to appear.

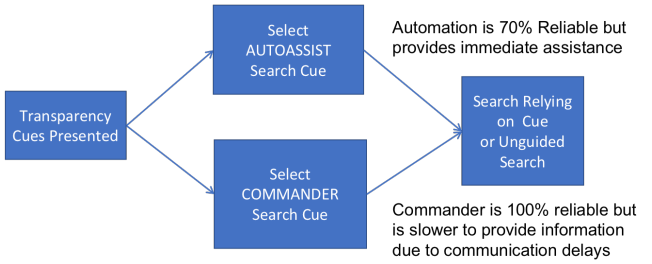


Figure 2: Diagram of the trial stages. A trial starts with presentation of transparency cues. Participants make a 2AFC search assist cue selection. Then they complete the visual search by either relying on the selected assist or searching unguided.

AUTOASSIST simulates automation to provide a search cue. Unlike the COMMANDER, it is available immediately at the start of the search stage. However, AUTOASSIST is only 70% reliable, meaning it correctly pointed to the objective on 70% of trials and to a random location on the others.

Automation Transparency Cues Transparency information was provided on every trial to aid participants in their assist type decisions. Participants could use the cues to learn when the AUTOASSIST would be unreliable. The two types of transparency cues were: the age of the data and number of sensors available to the automation. The number of sensors ranged from 1 to 3, and AUTOASSIST was unreliable if there was only 1 sensor. The data age varied from 1 to 36 hours old, and AUTOASSIST was unreliable if the data was over 24 hours old.

Fallon et al. (2019) used four transparency cue conditions. In the age-only condition, a statement about the age of the sensor data was given; no information about the number of sensors was provided. In the number-only condition, a statement about the number of sensors was given; no information about the age of the data was provided. In the age + number text condition, a statement included both the age and number information. And in the age + number graphic condition, the combination of age and number information was presented in a visual representation leveraging a circle-packing graphic.

Feedback An important component of modeling learning from experience is accounting for the feedback received about the outcome of one's decisions. We model two types of feedback received by participants during the search stage. The first was direct observation of success or failure of the AUTOASSIST. On trials when the AUTOASSIST was unreliable, it would visibly fail by disappearing from the screen at the moment of failure.

The second source of feedback was the total time to execute each search; participants were not given explicit timing information, but experienced how long each search took and

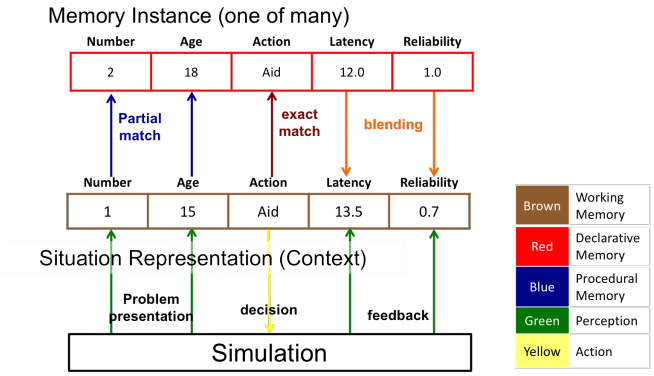


Figure 3: Diagram of the IBL model for the COBALT task decisions. The colors indicate the ACT-R mechanisms. See the text for a description of the instance representation. The upper half shows one trial as an instance in working memory (middle row), which is matched to similar prior instances in declarative memory (top row) through the matching and blending functions (blue, red and orange arrows). The lower portion of the diagram shows the perception and action mechanisms interacting with the task interface.

if multiple assist cues had to be selected to locate the objective (participants could request the COMMANDER after the AUTOASSIST failed, for example). The fastest search trials occurred when a participant selected a reliable AUTOASSIST and followed it directly to the objective. The slowest search trials occurred when a participant selected an unreliable AUTOASSIST, followed it and observed its failure, and then attempted some combination of unguided search, calling and waiting for the COMMANDER assist, and then relying on the COMMANDER assist to locate the objective. In this way, the negative feedback from long search times with unreliable automation might influence experience differently than the positive reinforcement and short search times associated with relying on reliable automation.

Human Data Sixteen participants completed the four transparency conditions (Age, Number, Age + Number Text, and Graphic) of the COBALT task, for a within-subjects manipulation. A single condition contained 13 blocks of 11 trials for a total of 572 trials per participant. Transparency condition orders were randomized between participants.

Cognitive Model of Automation Reliance

We developed a model of the automation reliance decisions in COBALT task with Instance-base Learning Theory (IBLT; Gonzalez & Dutt, 2011; Gonzalez et al., 2003) implemented in the ACT-R cognitive architecture (Anderson et al., 2004). IBL a methodology for modeling problem solving and decision making that relies on previous experiences rather than pre-defined strategies. Those experiences are stored in the declarative memory of the cognitive architecture, whose mechanisms support adaptive storage and associative retrieval. Experiences are stored in memory as a combination of situation features, decision taken, and observed outcome.

Memory instance availability is controlled by activation:

$$A_i = \log\left(\sum_{j=1}^{N} t_j^{-d}\right) \quad (1)$$

where i is the memory and d is the decay parameter controlling the power law of recency; the summation over all references to that memory provides the power law of practice. Given a particular situation, relevant memories are retrieved by computing their match score that combines their activation with their degree of relevance:

$$M_i = A_i + \sum_{j=1}^{l} MP \times Sim(d_j, v_{ij}) \quad (2)$$

where j is a feature in the situation representation, d_j is the corresponding value in the current situation, v_{ij} is the corresponding value in memory i, and *Sim* is the similarity between those two values. Rather than retrieving a single memory, a consensus outcome is generated using the memory blending mechanism satisfying this constraint:

$$V = \underset{V_j}{\operatorname{argmin}} \sum_{i=1}^{k} P_i \times Sim(V_j, v_{ij})^2 \quad (3)$$

where V is the consensus value among the set V_j of possible values, and P_i is the probability weight of memory i, reflecting its match score M_i through a Boltzmann softmax distribution.

Our IBL model adopts a straightforward representation of the problem. Examples are shown in Figure 3, where the middle row is a current trial instance, and the top row is one similar instance from declarative memory. The situation features are the age and/or number cues; the decision is whether to rely on the COMMANDER or AUTOASSIST (labeled *aid* in Figure 3), and the outcome is whether the AUTOASSIST was reliable (*Reliability*) and time to complete the visual search (*Latency*). To make a decision, the model generates an expected outcome for each assist type by performing blended retrievals for the specific situation feature(s) available (age, number, both) and each assist type, extracting an expected value for total search time. The model selects the assist type with the lowest expected search time. It then generates an expectation for the reliability of the automation in a similar manner, using a blended retrieval over situation feature(s) and selected assist type. The model then executes the option, and stores a new instance combining that situation's feature(s), the option chosen, and the outcomes experienced in terms of reliability and search latency. Finally, at the end of each condition, the model generated a general expectation of reliability through a blending retrieval with no features specified.

IBL models need either a back-up strategy (such as random exploration) to get started, or some initial instances to bootstrap the process. We chose the latter route, creating three instances to represent as broad a range of outcomes as possible. Those instances could have resulted from a short practice phase, or fairly straightforward reflection upon the instructions; both instructions and a few practice trials were given to

Table 1: Signal Detection Theory Mapping of Automation Reliance Intention Formations

	Actual Reliability of AUTOASSIST	
	Reliable	Unreliable
AUTOASSIST Selected	True Positive	False Positive
COMMANDER Selected	False Negative	True Negative

COBALT participants. The first instance featured the most reliable cues (3 sensors and 1-hour-old data), a decision to rely on AUTOASSIST, and outcomes of reliable AUTOASSIST and fastest search time (directed search time of 3 seconds). The second instance featured the least reliable cues (1 sensor and 36-hours-old data), a decision to rely on AUTOASSIST, and outcomes of unreliable AUTOASSIST, and the slowest search time (random search time of 15 seconds). The third instance featured average cues (2 sensors and medium age), a decision to rely on COMMANDER, and outcomes of reliability and an intermediate search time (wait then direct search for a total time of 8 seconds). We use ACT-R default parameters: decay $d = 0.5$; activation noise $s = 0.25$; mismatch penalty factor $MP = 1.0$; linear similarities over $[0, -1.0]$.

Results

We focus on three aspects of the data collected by Fallon and colleagues: decision stage intention formation choices, search stage automation reliance actions, and the subjective ratings of the AUTOASSIST's reliability. We consider together the human and model data. Our goal is to evaluate if the model captures well the human behaviors and if the IBL model's internal representation reflects trust calibration.

Predictive Power Metrics We quantify reliance calibration with predictive power analysis, based on a signal detection theory (SDT) characterization of automation use decisions (Feinstein, 1975). SDT quantifies the decision rates about the two assist types balanced with the ground truth of the AUTOASSIST's reliability. For the decision stage, we define a true positive as a decision to request AUTOASSIST when reliable and a false positive as a decision to request AUTOASSIST when it is unreliable. Table 1 defines all four SDT categories for the decision stage, reflecting intention formation accuracy, and Table 2 gives the definitions in the search stage for reliance actions accuracy.

We define positive predictive power:

$$\text{PPP} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}. \quad (4)$$

PPP gives the proportion of trials a participant appropriately chose AUTOASSIST out of all trials on which the participant selected the AUTOASSIST option. We define negative predictive power:

$$\text{NPP} = \frac{\text{True Negative}}{\text{True Negative} + \text{False Negative}}. \quad (5)$$

NPP gives the rate at which the participant appropriately did *not* select the AUTOASSIST (selected COMMANDER in the decisions stage or did not follow an unreliable search cue) when it would have been unreliable, out of all the trials on which the participant did not select AUTOASSIST.

We selected PPP and NPP as our metrics for appropriate reliance because they reflect the decision maker's ability to correctly select the automation when it will be reliable and not select the automation when it will be unreliable, respectively, while accounting for the prevalence of reliable and unreliable trials in the experiment. Accounting for the base rate of reliability is a core part of the definition of trust/reliance calibration. We note the more common SDT metrics d' and β (decision criterion) have been used in many studies to examine human judgements about the reliability of alarms or automation recommendations. These metrics emphasize the participants' abilities to discriminate signal cues from noise or non-signals. Application in the present study would measure participants' abilities to discriminate the transparency cues indicating the AUTOASSIST's reliability; the emphasis is on how participants internally represent the transparency cues. Understanding this internal representation is important for selecting effective transparency cues, but our present interests are more about quantifying decision makers' automation reliance, informed by those cues. PPP and NPP better serve this goal. Additionally, there is evidence that PPP and NPP better reflect the time-varying nature of decision-making processes without changing their statistical properties (Repperger, Warm, Havig, Vidulich, & Finomore, 2009).

Assist Selection Decisions Predictive Power Figure 4 (top) gives the predictive power for both the humans and models in the assist selection decision stage. The bars give the means, and the points are the individual decision makers. For the human decision makers, PPP and NPP are fairly high. PPP (right) is similar across all transparency cue conditions; NPP (left) shows a bit more variability, with the highest NPP observed in the Number-only condition. NPP for humans is comparable to their PPP. Between both metrics, we can infer that generally people chose the appropriate assist more often than the inappropriate one.

The model closely reproduced the average level of PPP in all conditions and NPP in the text and graphic (two-cue) conditions. The model underestimates NPP in the Age-only and Number-only conditions, meaning the model makes a higher number of false negative decisions than humans. This discrepancy might result from transfer between conditions, as the model currently makes the assumption that no transfer occurs across conditions because of distinct representations of situation features. It is possible that participants relied on information between conditions, improving performance on single cues, relative to the model lacking between-condition learning. Recent increases in representation flexibility in the

Table 2: Signal Detection Theory Mapping of Reliance on AUTOASSIST Search Cues

	Actual Reliability of AUTOASSIST	
	Reliable	Unreliable
AUTOASSIST Followed	True Positive	False Positive
AUTOASSIST Not Followed	False Negative	True Negative

ACT-R architecture enables us to explore alternative assumptions in future work.

Reliance on Search Assist Predictive Power The second way we quantify reliance on automation is to further use Equations 4 and 5 in the search stage to examine the proportion of trials wherein people followed the AUTOASSIST cue when it was or was not reliable. Table 2 summarizes the SDT definitions for AUTOASSIST search reliance actions. Here, we consider only the subset of trials on which participants selected AUTOASSIST in the decision stage, because there was no automation reliance action when COMMANDER was selected. PPP with the Table 2 mapping is the proportion of trials on which a participant followed a reliable AUTOASSIST out of all trials on which participants followed the AUTOASSIST; NPP is to the proportion of trials on which the participant did not follow the unreliable AUTOASSIST cue out of all trials on which they did not follow the AUTOASSIST.

Figure 4 (bottom) shows the distributions of PPP (right) and NPP (left) for the search stage of the COBALT trials. The means for PPP are higher than NPP, within each measure, the human means are similar across all conditions. The distributions for NPP have a larger variance, in addition to the lower means. The low (approximately .25) NPP means that the participants are taking more false negative reliance actions than true negatives. Compared to the decision stage (Figure 4 upper), the search NPP means are much lower; PPP distributions and means are similar in the two task stages.

In the search stage, the model generates expectations of the AUTOASSIST's reliability, which we translated into predictive power measures. The model qualitatively reproduced both PPP and NPP behaviors. We observe that the larger variability for NPP might result from individual differences in strategy, which we plan to explore in future work.

Perceived Reliability of the Automation An exciting result that emerges from the model is an estimate of the probability of overall automation reliability that appears to parallel the human subjective ratings.

At the completion of each condition, participants were asked to estimate the AUTOASSIST's correctness for that condition. Ratings were given as a value between 0 and 100%. The ground truth automation reliability was always 70%. Figure 5 gives the means and individual ratings from

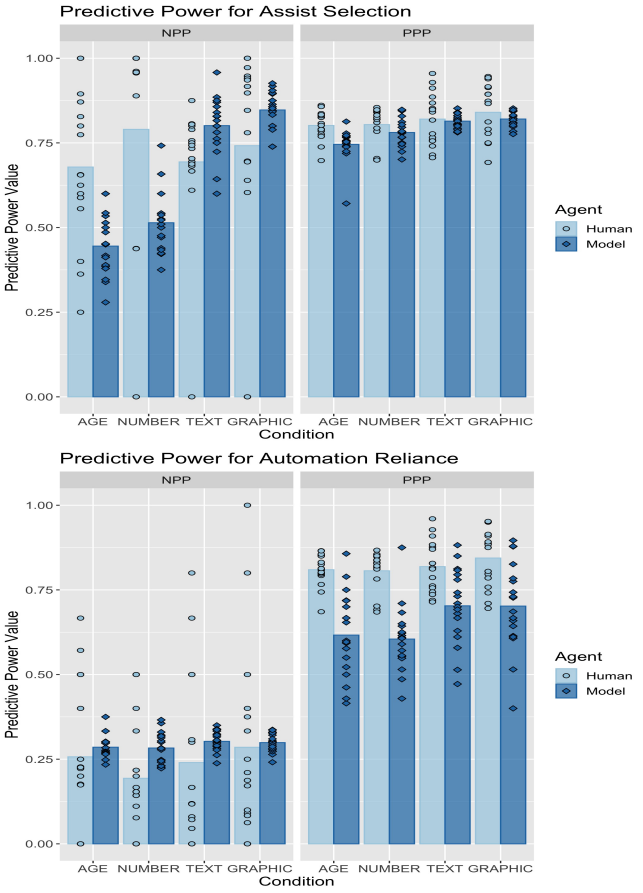


Figure 4: Distributions of predictive power values for all conditions for the decision stage (top) and search stage (bottom) of the COBALT trials.

both humans and IBL models; a horizontal line indicates the actual automation reliability. As shown in Figure 5, on average, both people and models over-estimated the AUTOASSIST's reliability. Over-estimations were larger in the text and graphic conditions than in the single cue conditions.

Previous efforts established the ability of IBL models to reproduce human cognitive biases resulting from the interaction of cognitive mechanisms and task statistics (Lebiere et al., 2013). This predictive basis for judgments of (over)trust raises the potential of using cognitive models to support human-machine teaming in ways that automatically compensate for human biases. Importantly, as conceptualized by definitions of trust calibration, internal estimates of reliability were shaped through reliance experiences.

Relationship to Conceptual Trust Calibration Model

Our IBL model's performance provides empirical support for the closed-loop dynamic calibration process of Lee and See's (2004) model. However, our process model does not yet integrate the moderating factors outlined in their conceptual model. Despite only formalizing the cognitive mechanisms in Lee and See's (2004) feedback loop, our approach was still

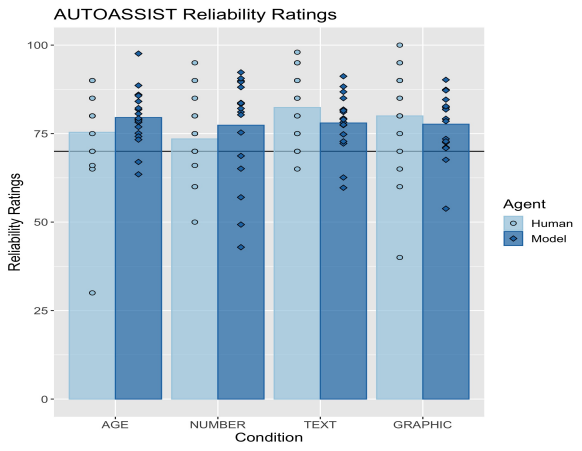


Figure 5: Mean perceived reliability ratings (bars) and individual reliability ratings (points) each decision maker. The light blue is the human subjective ratings data; the dark blue are reflect blended values in the IBL model. The horizontal line at 70 indicates the ground truth automation reliability.

able to fairly closely mimic human responses. IBL model performance suggests that the individual, organizational, cultural and environmental context played a less important role in influencing trust calibration within this controlled task environment. In some ways, these findings are not surprising because we attempted to control for (and did not manipulate) many of these variables. What is less clear from our findings is whether our model's ability to simulate trust calibration would generalize to other less constrained environments where individual, organizational, cultural and environmental context might be more influential. If they do, such findings would suggest that the feedback loop in the bottom portion of Lee See's model is the most powerful driver of trust calibration. Perhaps the experience gained from interacting with the automation has such a powerful impact on trust and reliance calibration that simply modeling this cycle is sufficient to replicate human trust dynamics. The impact of organization, culture environment and individual differences must be explored; the IBL model should allow for a systematic investigation into the impact of these variables on trust calibration.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036–1060.

Chen, J. Y., & Barnes, M. J. (2014). Human–agent teaming for multirobot control: A review of human factors issues. *IEEE Transactions on Human-Machine Systems*, *44*, 13–29.

de Visser, E. J., Peeters, M. M., Jung, M. F., Kohn, S., Shaw, T. H., Pak, R., & Neerincx, M. A. (2020). Towards a theory of longitudinal trust calibration in human–robot teams. *International Journal of Social Robotics*, *12*, 459–478.

Fallon, C. K., Blaha, L. M., Jefferson, B., & Franklin, L. (2019). A capacity coefficient method for characterizing the impacts of automation transparency on workload efficiency. In *Proceedings of the Human Factors and Ergonomics Society* (pp. 827–832).

Fallon, C. K., Murphy, A. K., Zimmerman, L., & Mueller, S. T. (2010). The calibration of trust in an automated system: A sensemaking process. In *2010 International Symposium on Collaborative Technologies and Systems* (pp. 390–395).

Feinstein, A. R. (1975). XXXI. On the sensitivity, specificity, and discrimination of diagnostic tests. *Clinical Pharmacology & Therapeutics*, *17*(1), 104–116.

Gonzalez, C., & Dutt, V. (2011). Instance-based learning: Integrating sampling and repeated decisions from experience. *Psychological Review*, *118*(4), 523–551.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*(4), 591–635.

Jessup, S. A., Schneider, T. R., Alarcon, G. M., Ryan, T. J., & Capiola, A. (2019). The measurement of the propensity to trust automation. In *International conference on human-computer interaction* (pp. 476–489).

Kramer, R. M. (1999). Trust and distrust in organizations: Emerging perspectives, enduring questions. *Annual Review of Psychology*, *50*(1), 569–598.

Lebiere, C., Pirolli, P., Thomson, R., Paik, J., Rutledge-Taylor, M., Staszewski, J., & Anderson, J. R. (2013). A functional model of sensemaking in a neurocognitive architecture. *Computational Intelligence and Neuroscience*, *2013*, 921695.

Lee, J. D., & Moray, N. (1994). Trust, self-confidence, and operators' adaptation to automation. *International Journal of Human-Computer Studies*, *40*(1), 153–184.

Lee, J. D., & See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human Factors*, *46*, 50–80.

Lyons, J. B., Koltai, K. S., Ho, N. T., Johnson, W. B., Smith, D. E., & Shively, R. J. (2016). Engineering trust in complex automated systems. *Ergonomics in Design*, *24*(1), 13–17.

Merritt, S. M., & Ilgen, D. R. (2008). Not all trust is created equal: Dispositional and history-based trust in human-automation interactions. *Human Factors*, *50*, 194–210.

Muir, B. M. (1987). Trust between humans and machines, and the design of decision aids. *International Journal of Man-Machine Studies*, *27*(5-6), 527–539.

Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human Factors*, *39*(2), 230–253.

Repperger, D., Warm, J., Havig, P., Vidulich, M., & Finomore, V. (2009). Modifying sensitivity/specificity for sensors using positive and negative predictive power measures. In *IEEE 2009 National Aerospace & Electronics Conference (NAECON)* (pp. 190–194).

Extending TransSet: An Individualized Model for Human Syllogistic Reasoning

Daniel Brand* **(daniel.brand@cognition.uni-freiburg.de)**
Nicolas Riesterer* **(riestern@cs.uni-freiburg.de)**
Marco Ragni (ragni@cs.uni-freiburg.de)
Cognitive Computation Lab, University of Freiburg
Georges-Koehler-Allee 52, 79110 Freiburg, Germany

Abstract

Recently, the TransSet model for human syllogistic reasoning was introduced and shown to outperform the previous state of the art in terms of predictive performance. In this article, we pick up the TransSet model and extend it to allow for capturing individual differences with respect to the conclusion "No Valid Conclusion" indicating that no logically correct conclusion can be derived from a problem's premises. Our evaluation is based on a coverage analysis in which a model's ability to capture individuals in terms of its parameters is assessed. We show that TransSet also outperforms state-of-the-art models on the basis of individuals and provide further evidence for the existence of an NVC aversion bias in human syllogistic reasoning.

Keywords: syllogistic reasoning; transset; modeling; transitivity

Introduction

Syllogistic reasoning is one of the longest-standing domains of reasoning research persisting for over a century now (for an early investigation, see Störring, 1908). Traditionally, a syllogistic problem consists of two quantified premises (*all*, *some*, *no*, *some ... not*) interrelating three terms (e.g., A, B, C):

All A are B
Some B are C

What, if anything, follows?

The goal in syllogistic reasoning is to relate the information conveyed by both premises via the middle term (B) occurring in both of them in order to draw a conclusion about the end terms (A, C). Depending on the arrangement of terms, a syllogistic problem is said to be in one of four figures (notation taken from Khemlani & Johnson-Laird, 2012):

Figure 1	Figure 2	Figure 3	Figure 4
A-B	B-A	A-B	B-A
B-C	C-B	C-B	B-C

By considering all combinations of quantifiers and term orderings, a total of 64 distinct syllogistic problems are obtained all of which can possibly be concluded by eight quantified relations between A and C in either direction, or "No Valid Conclusion" (NVC) indicating that no logically valid conclusion for the pair of premises exists. This results in a total of nine possible conclusions to each syllogistic problem.

Research in the domain of syllogistic reasoning quickly came to understand that human reasoners who are confronted with syllogistic tasks do not reason in accordance to classical first order logic but commit frequent and systematic errors which require psychological explanation (e.g., Woodworth & Sells, 1935; Wetherick & Gilhooly, 1995).

Since its early beginnings, the domain has inspired countless researchers to attempt to postulate and formalize assumptions about the processes underlying human syllogistic inference which has led to a wide variety of theories and models being introduced. In a meta-analysis (Khemlani & Johnson-Laird, 2012), a list consisting of the twelve most prominent theories of syllogistic reasoning was compiled and evaluated. The authors' analysis showed that because individual theories have their distinct strengths and weaknesses it is difficult if not impossible to identify a single best account.

More recently, TransSet (Brand et al., 2019), a model focusing on transitivity-based set interpretation, was introduced and shown to outperform the state-of-the-art models in terms of its predictive power on average human reasoning behavior. Still, in their discussion of TransSet's success, the authors highlighted the fact that a lot of potential for model performance remains untapped because most approaches currently do not account for the inter-individual differentiation underlying the wide variety of inferential strategies syllogisms are known to elicit (e.g., Roberts et al., 2001).

In this article, we attempt to push the TransSet model of syllogistic reasoning one step further by extending it to adapt to the behavior of individuals. By relying on findings from the syllogistic literature, we essentially integrate processing branches into the model which enable it to vary response strategies between individuals. We evaluate the resulting model based on a prediction task and compare its performance to both state-of-the-art models and statistical baselines to measure its success. Finally, we discuss our results as well as the implications of individualization for cognitive modeling research.

Related Work

The domain of syllogistic reasoning has extensively been approached from a multitude of directions including formal logics, probabilities, and various kinds of mental representations (for a review, see Khemlani & Johnson-Laird, 2012). However, in the last decade, the traditional model evaluation paradigm based on comparisons with group data obtained from experiments yielded results suggesting that model performances had reached a plateau making differentiation based on prediction accuracies difficult if not impossible (e.g., Bacon et al., 2003; Khemlani & Johnson-Laird, 2012).

More recently, a paradigm shift concerning the evaluation of models has started to gain traction. Inspired by theoretical and

*Both authors contributed equally to this manuscript.

empirical considerations of inter-individual differences (e.g., Molenaar, 2004) and the corresponding problem of group-to-individual generalizability (Fisher et al., 2018), the focus on model evaluation has shifted from aggregate representations of data to individual response data (e.g., Riesterer et al., 2019). Evaluating the state of the art in modeling human syllogistic reasoning in terms of predictions for individual response data revealed that previous analyses had overestimated model performances considerably. While Khemlani & Johnson-Laird (2012) reported values of up to 95%, 93%, and 84% for hits, correct rejections, and correct predictions on aggregate data, the comparison with individual responses showed that the best model only accounted for 34% of participants' responses (Riesterer et al., 2019). The new analysis produced two crucial results. First, overall low accuracies on participants' responses suggest that current models are far from what can possibly be considered accurate explanation of human behavior in syllogistic reasoning experiments. Second, comparisons with data-driven neural networks illustrated the considerable potential that remains in the domain especially when actively considering inter-individual differences.

A recent analysis (Riesterer et al., in press) put the focus of attention on a different aspect of individualized modeling: model parameterization. A *coverage* task was introduced in which models are fitted to individual response patterns and assessed in terms of their ability to reproduce the observed behavior from their latent parameterization. Computing the accuracy of the fitted models in comparison with the originally observed data allows to derive a score that enables a parameterization-centric assessment of individualized model performance. The analysis included two of the most prominent models for syllogistic reasoning, mReasoner (Johnson-Laird & Khemlani, 2013) and the Probability Heuristics Model (PHM; Chater & Oaksford, 1999). Briefly summarized, mReasoner is an instance of the mental model theory (e.g., Johnson-Laird, 1983) which assumes that individuals reason by constructing mental representations of the premises from which conclusion candidates are generated and potentially revised via a search for counterexamples. PHM, on the other hand, assumes that individuals reason in accordance to probabilistic validity as opposed to logic validity and postulates a set of heuristics to simulate this behavior.

The coverage evaluation (Riesterer et al., in press) revealed that both models are lacking in their ability to account for individual behavior. Only PHM managed to outperform the statistical baseline computed from the most-frequent answer (MFA; the optimal strategy for aggregate models in this task) and thereby demonstrated a basic albeit unimpressive ability to accommodate for individual reasoning behavior in terms of its parameterization. Overall, the coverage analysis highlighted the need for an increased focus on individual differences from a different perspective than the previous prediction-oriented analyses.

The TransSet Model

TransSet (Brand et al., 2019) is a recently introduced model for syllogistic reasoning which was developed with a different goal in mind than previous models. The current state of the art has largely originated from attempts at finding comprehensive explanations of human reasoning behavior which indirectly assumes the existence of general syllogistic inference processes available to all reasoners. However, because of empirical evidence about the variety of strategies employed in the syllogistic domain (e.g., Roberts et al., 2001) this assumption has been met with skepticism in the past (e.g., Bacon et al., 2003). TransSet acknowledges the existence of distinct inferential strategies and focuses on a specific type of naive reasoner who is untrained in the task of solving syllogistic problems and therefore relies on intuitive reasoning based on the prominent surface features of syllogisms, i.e., quantifiers and term order. In particular, it expects reasoners to rely on the general concept of transitivity because of its relevance and importance in everyday reasoning (e.g., for argumentation). In doing so, TransSet reflects a single-strategy model that uses the surface features of syllogisms (e.g., quantifiers or the order of terms) to derive its predictions. Its inferential mechanisms are built on the assumption that reasoning can be defined on the basis of a set-based interpretation of premises and a transitivity-based inference scheme.

TransSet generates predictions for syllogistic problems based on a two-step process consisting of phases for conclusion direction and quantifier selection. The *direction selection phase* depends on the arrangement of terms in the premises. If the premises directly define a transitive path between the end terms (i.e., A-B;B-C or B-A;C-B), TransSet uses the positions of the end terms in the paths as the direction of the conclusion. Otherwise, it is assumed that reasoners attempt to modify the premises in order to create a transitive path. This is done by reversing one of the premises containing a universal quantifier, i.e., "All" or "No", with a preference for "All". If this is not possible, either because there is no universal quantifier or because of ties when both quantifiers are equal, NVC is returned aborting the inferential process.

The *Quantifier selection phase* uses the transitive path to infer the conclusion quantifier. The general assumption behind this phase is that individuals propagate information along the path. If the first quantifier is affirmative, both quantifiers are combined in accordance to the Atmosphere hypothesis (Woodworth & Sells, 1935). If the first quantifier is negative, information propagation is not possible directly. Here, TransSet assumes that if the second quantifier is "All", the disrupted flow of information along the path can be recovered by substituting the middle term with the last term on the path resulting in a "No" conclusion. If this is not possible, TransSet predicts NVC.

Crucially, the inferential mechanism proposed by TransSet does not incorporate traditional processes for deliberative reasoning (e.g., logics or the construction of mental representations) but focuses on a restricted mapping from syllogistic problems to specific conclusion predictions based on surface-features alone. As such, TransSet does not follow the goal to be an adequate explanation of the general behavior of human reasoners but assumes the existence of a subset of reasoners which follow the nonlogical (e.g., Evans, 1972) procedures it assumes. Still, it could be shown that TransSet outperforms the existing state of the art by a substantial margin (Brand et al., 2019). This does not necessarily mean that the processes assumed by TransSet are representative of the cognitive processes driving human inference. However, they currently give the best account of the data. It would be premature to consider TransSet generally superior to other models in its current state. Still,

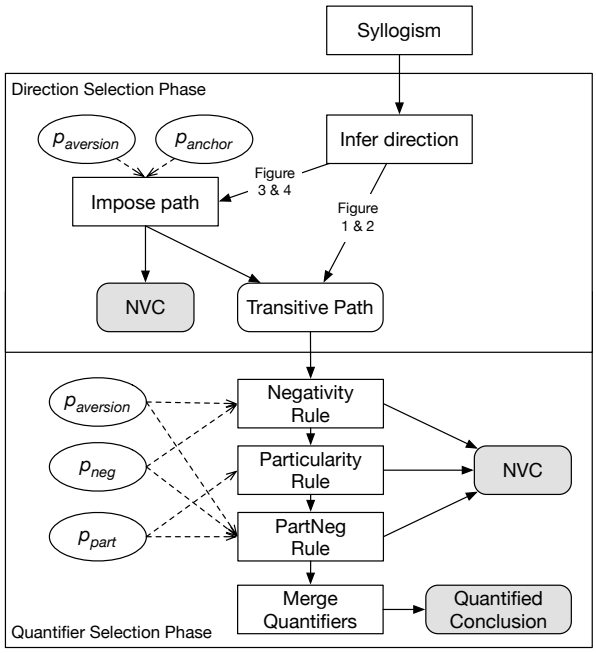


Figure 1: Overview over the individualized TransSet model.

it is a wake-up call for the proponents of the prevailing theories to justify their approaches on the level of response predictions.

Even though TransSet focuses on a single inferential strategy, it offers much potential for alteration with respect to the integration of individual differences. One of them is its handling of the NVC conclusion. Currently, NVC is generated either in cases where transitive paths cannot be formed or if information cannot be propagated along the path. However, research in syllogistic reasoning has produced two important findings with respect to NVC conclusions. First, response data suggests that reasoners are biased against producing NVC conclusions resulting in what could be considered an NVC aversion bias (e.g., Dickstein, 1976; Roberts et al., 2001). Although the reasons for this NVC aversion have not been conclusively determined as of yet, it seems reasonable to assume that individuals differ with respect to the influence it has on their reasoning making NVC aversion a promising component of an individualized model. Second, recent research has shown that certain forms of syllogism might invite NVC responses which suggests that NVC handling of individuals might not just simply be inhibited by aversion biases but also encouraged by certain combinations of premises (e.g., Galotti et al., 1986; Riesterer et al., 2020). As such, in the following sections we introduce and evaluate an extension of TransSet in terms of its NVC handling.

Individualizing TransSet

Our individualization which is summarized in Figure 1 focuses on NVC which is a peculiar conclusion in the syllogistic domain for various reasons. On the one hand, the NVC response itself might be ambiguous. Besides its intended meaning as an indication that no conclusion follows logically from the premises, NVC might also be interpreted as "giving up" signaling that a participant failed to arrive

Table 1: Parameter configurations and preconditions for NVC rules in the quantifier selection phase where Q_1 and Q_2 refer to the first and second quantifier of the transitive path, respectively.

Rule	$p_{aversion}$	p_{neg}	p_{part}	Precondition
Negativity	None	true	-	Q_1 negative
	Low	true	-	Q_1 negative, Q_2 not All
	High	true	-	Q_1,Q_2 negative
Particularity	-	-	true	Q_1,Q_2 particular
PartNeg	None	true	true	Q_1,Q_2 not All

at a quantified conclusion (Ragni et al., 2019). This interpretation can be an incentive for reasoners to "try harder" to avoid NVC responses which effectively invites illogical behavior. On the other hand, it is the logically correct answer for 37 out of the 64 problems (58%), i.e., for the majority of the domain. As there are nine possible conclusions, this imbalance might be unintuitive for some reasoners, especially since it is also unusual for riddles or puzzles, which the experimental setting might seem similar to, to be "unsolvable". This could lead to the NVC aversion phenomenon which has been discussed before (Roberts et al., 2001). However, the over-representation of NVC might also encourage the use of simple rules that can quickly derive an NVC response (e.g., Galotti et al., 1986).

One of the main concepts of TransSet is the separation of the deduction process into the direction selection and quantifier selection phases which each provide rules to check if the respective goals can be achieved. If any phase fails, NVC is concluded. However, the available rules and the likelihood to abort a phase may differ between individual reasoners which is why using them as starting points for the adaption to individual reasoners seems promising.

To allow TransSet to capture the effects of NVC, we introduced four parameters: $p_{aversion}$, p_{anchor}, p_{part}, and p_{neg}. The first parameter, $p_{aversion}$, represents the susceptibility to the NVC aversion bias of a reasoner (e.g., Dickstein, 1976) with possible values in [None, Low, High]. The parameter is used in both phases and determines the likelihood of accepting NVC responses. When NVC aversion is high, the phases of TransSet are less likely to fail since participants try to find a way around responding with NVC. For the direction selection phase, this means that a direction has to be selected, even if it is not clear if the conclusion should relate the end terms from A to C or vice versa (which can only occur for Figure 3 and Figure 4 syllogisms). In these cases, it is assumed that individual preferences decide if the end-term read first (A) is selected as an anchor point (resulting in the direction $A \rightarrow C$) or if the most recent term (C) is chosen (resulting in the direction $C \rightarrow A$). This preference is captured by the parameter p_{anchor} using the values [most-recent, first] which reflect the choice of anchor term. Note, however, that p_{anchor} is a conditional parameter which will only take effect when $p_{aversion}$ is high.

TransSet's second phase, in which the conclusion quantifier is determined, originally only had a single rule to derive NVC: When a transitive path starts with a negative quantifier ("No", "Some ... not"), the propagation of information along the path is prevented, which result in an NVC response in most cases (Brand et al., 2019). In this work, we extend the existing rule, allowing for several nuances depending on the aforementioned $p_{aversion}$ parame-

ter. In doing so, we integrate additional rules to derive NVC which have recently been shown to improve predictive performance when incorporated into various state-of-the-art models for syllogistic reasoning (Riesterer et al., 2020). In particular, we incorporate the rules related to negative quantifiers, i.e., *EmptyStart* and *Negativity*, into TransSet's original process handling negativity in the quantifier determination phase. The $p_{aversion}$ parameter is used to either strengthen or weaken the precondition starting from TransSet's original rule, which corresponds to $p_{aversion} = low$. The remaining rules proposed by Riesterer et al. (2020), i.e., *Particularity* and *PartNeg*, are integrated as additional rules. Individual availability of the above-mentioned rules is controlled via two binary parameters, p_{neg} and p_{part} which can either be set to true or to false. Table 1 summarizes the rules with the respective parameter configurations and the preconditions, that need to be fulfilled to derive NVC responses.

The proposed parameterization of TransSet is a natural extension of its original account. To match the behavior of the original TransSet model, the $p_{aversion}$ needs be set to "low" with p_{neg} being set to "true". Since $p_{aversion}$ is "low", the determination of the direction fails (resulting in a NVC response), which means that p_{anchor} has no effect. As a dedicated rule to derive NVC based on the particularity was not part of the original model, p_{part} needs to be set to "false".

It is important to note that all introduced parameters are categorical, i.e., rely on discrete values with all parameters except for $p_{aversion}$ being binary. While continuous parameter values are generally useful to describe the probabilities or relative importances of effects occurring in populations, a deterministic model with discrete parameters is a preferable description of individuals. Since the data only represents a snapshot of an individual's reasoning behavior, we cannot derive probabilities for their decisions (especially if each syllogistic problem was only solved once per individual). On an individual level, probabilities are only meaningful if each individual repeatedly provided responses to the same tasks. Thus, we have to resort to evaluating the ability of a model to reproduce exact patterns which naturally suggests deterministic model behavior and parameter usage.

Method

The core objective of the following analysis is to evaluate our extension of TransSet in terms of its ability to account for the inferential behavior of individual human reasoners. To this end, we rely on a coverage task (Riesterer et al., in press) in which the goal is to capture the response behavior of individuals in the model's parameters. By assessing the residual error, an estimate of the model's ability to account for inter-individual differences is obtained. Additionally, the parameter configurations resulting from fitting the model to individuals allows for an interpretation of the variation in the observable reasoning behavior in terms of the processes assumed by the model.

Coverage Analysis Setting

Our analysis focuses on evaluation TransSet's ability to recover individual reasoning behavior from its latent parameterization. Put differently, we assess the degree to which TransSet's parameter space *covers* individuals (Riesterer et al., in press).

Note, that the justification of coverage analyses depends on the models being included. In the case of database-like models which fit by storing the observed information, coverage will always be perfect since behavior can simply be recalled from the database. In the case of cognitive models, however, parameters usually have an associated meaning and try to capture essential properties of the assumed mental processes. As such, coverage gives a meaningful estimate of a model's ability to accommodate for individuals.

To increase the expressiveness of our analysis by providing a reference frame for the obtained coverage scores, we include the models from the previous coverage analysis (Riesterer et al., in press): mReasoner (Johnson-Laird & Khemlani, 2013) and the Probability Heuristics Model (Chater & Oaksford, 1999), as well as a random uniform model and the *Most-Frequent Answer* (MFA) model which generates predictions based on the most frequently observed response to a syllogistic problem in a dataset.

Dataset & Implementation

For our analysis, we rely on the *Cognitive Computation for Behavioral Reasoning Analysis* (CCOBRA) framework[1] for model evaluation. The dataset we use is the "Ragni2016" dataset for syllogistic reasoning which is openly available as part of the framework and has been used as benchmark data in many evaluations of syllogistic models including the previously introduced coverage analysis (Riesterer et al., in press). It consists of a total of 139 participants who were presented with the full set of 64 syllogisms and asked to select which of the nine possible responses followed from syllogistic premises. The data, model implementation, and analysis scripts developed for this article are available on GitHub[2]

Analysis & Results

Performance Analysis

Figure 2 depicts the results of the coverage evaluation obtained from CCOBRA. The box plots provide a descriptive view of coverage scores, i.e., the models' abilities to reconstruct reasoning behavior from their parameterizations, while the dots represent the scores for the 139 individuals from the dataset. When fitted to individual reasoners, TransSet significantly improves over the original model (median coverage scores of 0.50 and 0.44, respectively; Mann-Whitney U Test, $U = 7783.5$, $p = .0025$) and substantially outperforms mReasoner and PHM (median coverage scores of 0.38 and 0.45, respectively). TransSet and PHM also surpass the performance of the MFA (median coverage score of 0.45), which is the upper bound for models disregarding individual differences, showing that the concepts underlying their parameters are suited to capture the behavior of individuals. However, it is important to note that TransSet only describes a specific strategy that some individuals might use. When considering the results for specific individuals, it becomes apparent that a substantial amount is still not sufficiently covered by the model. While this might partially be due to guessing-like behavior or non-systematic mistakes, it also possible that some of these individuals are using different strategies.

The general improvement of TransSet achieved by our individualization indicates that the incorporation of NVC biases is a promising way for models to account for different individuals. This is not

[1] https://github.com/CognitiveComputationLab/ccobra
[2] https://github.com/Shadownox/iccm-transset-indiv

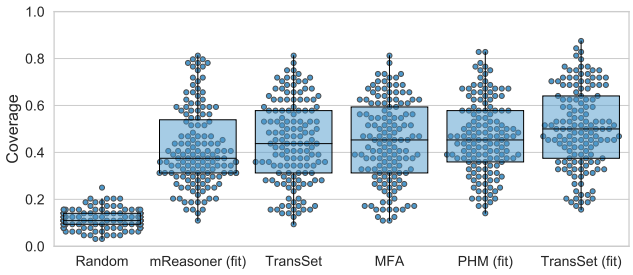


Figure 2: Accuracies of models for the coverage task. The suffix "(fit)" indicates that the model was fitted to an individual's responses.

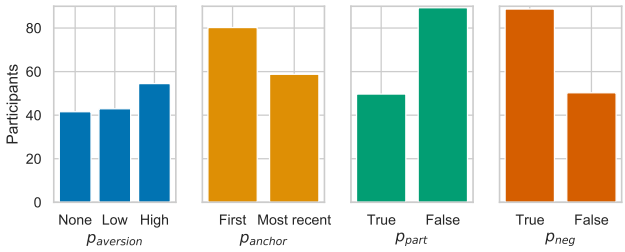


Figure 3: Parameter distributions for the individualized TransSet's parameters resulting from fitting the model to individuals from the "Ragni2016" dataset.

surprising, given the special status of the NVC due to the imbalance of the syllogistic task and ambiguity of the response. On a more general level, the improvement also shows that it is possible to significantly boost the performance of a model by focusing on rules and mechanisms that are able to differentiate between individuals. This highlights the importance of an evaluation based on individual data, as improvements beyond the performance of the MFA cannot be assessed on the basis of aggregated analyses. Additionally, these analyses provide a starting point for further investigations of individuals that are not covered sufficiently by most state-of-the-art models. For now, it is unknown to which extent this can be attributed to noise (e.g., due to guessing-like behavior). An in-depth analysis of these individuals might help to better estimate the proportion of noise and uncover additional strategies and biases in the data.

Parameter Distribution Analysis

Apart from allowing models to be fitted to individual reasoners, the utilization of parameters is also an important property of models based on which their internal integrity can be assessed. Ideally, parameters in cognitive models have specific meaning in relation to the assumed inference process. For instance, in the case of TransSet, the NVC aversion parameter $p_{aversion}$ is indicative for the model's behavior to pursue alternative conclusions to avoid NVC. Optimally, the use of parameters in cognitive models should be limited to the minimum that still enables the capturing of distinct and important differences between individuals (principle of parsimony). This, in turn, means that all parameter assignments should be relied on by the model to account for a population of reasoners. If certain parameter configurations are only used for negligible amounts of individuals, either the corresponding group of individuals was not part of the data or, more likely, the model has an inefficient use of parameters and should be revised in order to reduce its parameter complexity and increase its explainability.

Figure 3 shows the distributions for TransSets's parameters. For each possible value of a parameter, the number of participants that are described best by using the respective value is shown. When analyzing the distribution for $p_{aversion}$, we see that $p_{aversion} = high$ yields the best results for the majority of individuals, indicating that incorporating NVC aversion is indeed beneficial for individualized models of syllogistic reasoning. The particularity rule, despite being inactive for the majority of participants, still seems to be a valuable addition, as it still improved the fit for a third of the individuals. While the optimal parameterization for the majority of the data has the biggest importance for the fit, all parameter configurations still represent a substantial number of individuals. In the case of $p_{aversion}$, the majority of participants is not even represented by the most prominent value ("high"). The original TransSet model corresponds to $p_{aversion} = low$, which does not reflect the NVC aversion of most individuals, but instead describes the data better on an aggregate level. This highlights the importance of individual modeling in general: A model describing the average reasoner might not be able to reflect the most prevalent traits of reasoning.

With respect to the hypothesis of an aversion against NVC, the distribution of $p_{aversion}$ is intriguing. A substantial number of participants are described by high NVC aversion, leading to response patterns with little to no NVC responses. However, while this group does in fact seem to avoid NVC wherever possible, the majority of participants have a low or no NVC aversion at all. Therefore, the aversion against NVC conclusions seems to be a highly individual behavior that affects a substantial proportion of the participants but might not necessarily be a universal factor in human syllogistic reasoning behavior. Since only a group of individuals seems to avoid the NVC response consistently, this hints at general misunderstandings of the NVC response itself for this group.

General Discussion

In this article, we presented and evaluated an individualization of the recently introduced TransSet model for syllogistic reasoning (Brand et al., 2019). To integrate the capability to differentiate between individuals we focused our attention on the conclusion "No Valid Conclusion" (NVC) which has been in the focus of attention before for its ability to evoke aversion biases (e.g., Dickstein, 1976; Roberts et al., 2001) and for being a conclusion which was neglected by a number of models in the past (Riesterer et al., 2020).

TransSet's original specification already contained rules to derive NVC conclusions directly from surface features of syllogistic premises which were invoked when the construction of transitive paths or the propagation of information along them failed (Brand et al., 2019). Our individualization of the model extends on these rules by introducing parameters with the goal to capture individual differences in NVC behavior. We assume a total of four parameters reflecting (1) the magnitude of the aversion against NVC responses, (2) a figure anchor providing the direction of the conclusion

generated in alternative to NVC following the NVC aversion, and the susceptibility to directly conclude NVC based on (3) negativity or (4) particularity of the premises (Riesterer et al., 2020).

Our results illustrate the success of TransSet's individualization. In a coverage analysis (for an introduction to the paradigm, see Riesterer et al., in press), the model is shown to outperform not only the statistical model following a response strategy focusing on the conclusions most frequently selected by participants (most-frequent answer, MFA), but also the two state-of-the-art models mReasoner (Johnson-Laird & Khemlani, 2013) and the Probability Heuristics Model (PHM; Chater & Oaksford, 1999) which have been separately analyzed in a coverage analysis by Riesterer et al. (in press). Investigating the parameter distribution that follows from fitting TransSet to individuals illustrates the quality of the assumed factors for individualization. The parameter space is evenly distributed with no value being only assigned to a negligible number of participants. Further, the distribution of the aversion parameter adds to the evidence for such a bias in syllogistic reasoning (e.g., Dickstein, 1976; Ragni et al., 2019).

Overall, our results add to the growing corpus of modeling research on the level of individual responses. Despite the fact that TransSet is intended to only capture a distinct subset of reasoners, namely those who rely on surface-level features of the problem domain (e.g., quantifiers and term order), it currently outperforms even the most comprehensive and general models of the state of the art both on the aggregate and individual level. While we should refrain from considering it an overall superior explanation of human cognition in this task, especially given its current lack of grounding in terms of psychological/neuroscientific concepts, it should serve as a wake-up call to theorists and modelers alike. Our results demonstrate that the previous signs of a performance-based plateau were merely due to the choice of a severely restricted evaluation paradigm which can be overcome by adopting the perspective of individual responses.

Acknowledgements

This paper was supported by DFG grants RA 1934/2-1, RA 1934/3-1 and RA 1934/4-1 to MR.

References

Bacon, A., Handley, S., & Newstead, S. (2003). Individual differences in strategies for syllogistic reasoning. *Thinking & Reasoning*, *9*(2), 133–168.

Brand, D., Riesterer, N., & Ragni, M. (2019). On the matter of aggregate models for syllogistic reasoning: A transitive set-based account for predicting the population. In T. Stewart (Ed.), *Proceedings of the 17th International Conference on Cognitive Modeling* (pp. 5–10). Waterloo, Canada: University of Waterloo.

Chater, N., & Oaksford, M. (1999). The probability heuristics model of syllogistic reasoning. *Cognitive Psychology*, *38*(2), 191–258.

Dickstein, L. S. (1976). Differential difficulty of categorical syllogisms. *Bulletin of the Psychonomic Society*, *8*(4), 330–332.

Evans, J. S. B. T. (1972). On the problems of interpreting reasoning data: Logical and psychological approaches. *Cognition*, *1*(4), 373–384.

Fisher, A. J., Medaglia, J. D., & Jeronimus, B. F. (2018). Lack of group-to-individual generalizability is a threat to human subjects research. *Proceedings of the National Academy of Sciences*, *115*(27), E6106–E6115.

Galotti, K. M., Baron, J., & Sabini, J. P. (1986). Individual differences in syllogistic reasoning: Deduction rules or mental models? *Journal of Experimental Psychology: General*, *115*(1), 16–25.

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness*. Cambridge, MA: Harvard University Press.

Johnson-Laird, P. N., & Khemlani, S. S. (2013). Toward a unified theory of reasoning. In *Psychology of learning and motivation* (pp. 1–42). Elsevier.

Khemlani, S., & Johnson-Laird, P. N. (2012). Theories of the syllogism: A meta-analysis. *Psychological Bulletin*, *138*(3), 427–457.

Molenaar, P. C. M. (2004). A manifesto on psychology as idiographic science: Bringing the person back into scientific psychology, this time forever. *Measurement: Interdisciplinary Research & Perspective*, *2*(4), 201–218.

Ragni, M., Dames, H., Brand, D., & Riesterer, N. (2019). When does a reasoner respond: Nothing follows? In A. K. Goel, C. M. Seifert, & C. Freksa (Eds.), *Proceedings of the 41st Annual Conference of the Cognitive Science Society* (pp. 2640–2646). Montreal, QB: Cognitive Science Society.

Riesterer, N., Brand, D., Dames, H., & Ragni, M. (2020). Modeling human syllogistic reasoning: The role of "No Valid Conclusion". *Topics in Cognitive Science*, *12*(1), 446–459.

Riesterer, N., Brand, D., & Ragni, M. (2019). Predictive modeling of individual human cognition: Upper bounds and a new perspective on performance. In T. Stewart (Ed.), *Proceedings of the 17th International Conference on Cognitive Modeling* (pp. 178–183). Waterloo, Canada: University of Waterloo.

Riesterer, N., Brand, D., & Ragni, M. (in press). Do models capture individuals? Evaluating parameterized models for syllogistic reasoning. In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society*.

Roberts, M. J., Newstead, S. E., & Griggs, R. A. (2001). Quantifier interpretation and syllogistic reasoning. *Thinking & Reasoning*, *7*(2), 173–204.

Störring, G. W. (1908). Experimentelle Untersuchungen über einfache Schlussfolgerungsprozesse. *Archiv für die gesamte Psychologie*, *11*, 1–127.

Wetherick, N. E., & Gilhooly, K. J. (1995). 'Atmosphere', matching, and logic in syllogistic reasoning. *Current Psychology*, *14*(3), 169–178.

Woodworth, R. S., & Sells, S. B. (1935). An atmosphere effect in formal syllogistic reasoning. *Journal of Experimental Psychology*, *18*(4), 451–460.

Reinforcement Learning for Production-based Cognitive Models

Adrian Brasoveanu (abrsvn@gmail.com)
Department of Linguistics, 1156 High Street
Santa Cruz, CA 95064, USA

Jakub Dotlačil (j.dotlacil@gmail.com)
Utrecht University
Utrecht, The Netherlands

Abstract

We introduce a framework in which we can start exploring in a computationally explicit way how complex, mechanistically specified, and production-based cognitive models of linguistic skills, e.g., ACT-R based parsers, can be acquired. Cognitive models for linguistic skills pose this learnability problem much more starkly than models for other 'high-level' cognitive processes, as they call for richly structured representations and complex rules that require a significant amount of hand-coding. In this paper, we focus on how Reinforcement Learning (RL) methods can be used as a way to solve the production selection problem in ACT-R. Production rules are treated as the actions of an RL agent, and the ACT-R model/mind as the environment in the RL sense. We focus on a basic learning algorithm (tabular Q-learning) and a very simple task, namely lexical decision (LD), framed as a sequential decision problem, with the goal of learning when a specific rule should be fired. Learning is faster and less noisy for shorter LD tasks (fewer stimuli), but the Q-learning agent manages to learn longer tasks fairly well. Realistically long LD tasks and more complex models, e.g., parsers, are left for future research.

Keywords: ACT-R, Reinforcement Learning, production-based models, linguistic skills, lexical decision

Learnability of production-based models

We introduce a framework in which we can start exploring in a computationally explicit way how complex, mechanistically specified cognitive models of linguistic skills, e.g., the parsers in Engelmann (2016); Hale (2011); Lewis and Vasishth (2005), can be acquired. Linguistic cognitive model learnability is an understudied issue, primarily because computationally explicit cognitive models are only starting to be more widely used in psycholinguistics. Cognitive models for linguistic skills pose this learnability problem much more starkly than models for other 'high-level' cognitive processes, since cognitive models that use theoretically-grounded linguistic representations and processes call for richly structured representations and complex rules that require a significant amount of hand-coding.

The learnability problem for production-rule based models can be divided into two parts:

i. rule acquisition – forming complex rules out of simpler ones, and
ii. rule ordering – deciding which rule to fire when.

ACT-R's (Anderson & Lebiere, 1998) partial answers to these problems are production compilation for (i), and rule-utility estimation for (ii). Apart from Taatgen and Anderson (2002), which investigates the role of production compilation in morphology acquisition, neither solution has been systematically applied to complex models for linguistic skills.

We focus here on the easier problem (ii). Our main contribution is to show how advances in the machine learning subfield of Reinforcement Learning (RL, Sutton & Barto, 2018) can be leveraged to solve it. RL and ACT-R have very close connections (Fu & Anderson, 2006), but they have remained largely unexplored.

Learning goal-conditioned rules: An example

The framework and the range of issues that emerge when we try to systematically integrate ACT-R and RL are best showcased with a simple example. We choose a basic learning algorithm, specifically, a tabular Q-learning agent, which is a model-free off-policy learning algorithm (Watkins, 1989; Watkins & Dayan, 1992). Also, we focus on a very simple task, namely lexical decision (LD). In an LD task, participants see a string of letters on a screen. If the participants think the string of letters is a word, they press one key (`J` in our setup); if they think the string is not a word, they press a different key (`F` in our setup). After pressing the key, the next stimulus is presented.

We investigate the extent to which the Q-learning agent can be used to learn goal-conditioned rules in an ACT-R based cognitive model of LD tasks. The main point of proposing and examining an ACT-R model of LD tasks is to construct a simple example of a production-rule based model that enables us to study learnability issues, and that can be scaled up in future work to more complex and cognitively realistic syntactic and semantic parsing models. In particular, the model provides the basic scaffolding of production rules needed for LD tasks, which is all that we need for our purposes. Fleshing the model out to capture major experimental results about LD, or comparing it to previously proposed cognitive models of LD is not our focus here.

LD tasks can be modeled in ACT-R with a small number of rules (see Brasoveanu and Dotlačil 2019 and Chapter 7 in Brasoveanu and Dotlačil 2020 for recent attempts), so they are a good starting example. We model three LD tasks of increasing length, hence difficulty:

i. 1 stimulus: the word *elephant*,
ii. 2 stimuli: the word *elephant* and a non-word, and

iii. 4 stimuli: the word *elephant*, a non-word, the word *dog*, and another non-word.

The model components are split between declarative memory, which stores the lexical knowledge of an English speaker, and procedural memory, which stores rules that enable the model to carry out the LD task. The rules are conditionalized actions: they fire/execute actions when their conditions are satisfied by the cognitive state of the ACT-R mind (the buffers). We assume 4 rules, provided in standard ACT-R format in (1)-(4) below (see Chapter 2 in Brasoveanu and Dotlačil 2020, for example, for more discussion of the format). These rules were originally hand-coded to fire serially by conditioning all the actions on specific goal states. The goal conditions are stricken out below because we remove them and let the Q-learning agent learn them.

(1) **Rule 1: Retrieving**

~~goal>~~ | ~~STATE:~~ ~~retrieving~~ |

visual> | VALUE: =val
VALUE: ∼FINISHED |

$\Longrightarrow$

goal> | STATE: retrieval_done |

+retrieval> | ISA: word
FORM: =val |

(2) **Rule 2: Lexeme Retrieved**

~~goal>~~ | ~~STATE:~~ ~~retrieval_done~~ |

retrieval> | BUFFER: full
STATE: free |

$\Longrightarrow$

goal> | STATE: retrieving |

+manual> | CMD: press-key
KEY: J |

(3) **Rule 3: No Lexeme Found**

~~goal>~~ | ~~STATE:~~ ~~retrieval_done~~ |

retrieval> | BUFFER: empty
STATE: error |

$\Longrightarrow$

goal> | STATE: retrieving |

+manual> | CMD: press-key
KEY: F |

(4) **Rule 4: Finished**

~~goal>~~ | ~~STATE:~~ ~~retrieving~~ |

visual> | VALUE: FINISHED |

$\Longrightarrow$

goal> | STATE: done |

With fully specified, hand-coded rules, the LD task unfolds as follows. Assume the initial goal STATE of the ACT-R model is `retrieving`, and the word *elephant* appears on the virtual screen of the model, which is automatically stored in the VALUE slot of the visual buffer.

At this initial stage, the preconditions of **Rule 1** are satisfied, so the rule fires. As a consequence, we attempt to retrieve a word with the form *elephant* from declarative memory, and the goal STATE is updated to `retrieval_done`. When the word is successfully retrieved, **Rule 2** fires and the `J` key is pressed. At that point:

i. in the 1-stimulus task, the text FINISHED is displayed on the screen, then **Rule 4** fires and ends the task;

ii. in the 2-stimuli task, the non-word is displayed, then **Rule 1** fires again; the retrieval attempt fails since we cannot retrieve a non-word from declarative memory, so **Rule 3** fires and the `F` key is pressed; at that point, the text FINISHED is displayed on the screen, then **Rule 4** fires and ends the task;

iii. in the 4-stimuli task, the first non-word is displayed, **Rule 1** fires again, then, just as in the 2-stimuli task, **Rule 3** fires and the `F` key is pressed, after which the word *dog* is displayed, **Rule 1** fires for the third time followed by **Rule 2**, which means that the `J` key is pressed and the second non-word is displayed; now, **Rule 1** fires for the final time, followed by **Rule 3**, which triggers the `F` key to be pressed; at this point, the 4-stimuli task is over, so the text FINISHED is displayed on the screen, then **Rule 4** fires and ends the task.

Thus, assuming fully specified, hand-coded rules, the sequences of rule firings for the three LD tasks are as follows:

i. 1-stimulus task: Rules **1 – 2 – 4**

ii. 2-stimuli task: Rules **1 – 2 – 1 – 3 – 4**

iii. 4-stimuli task: Rules **1 – 2 – 1 – 3 – 1 – 2 – 1 – 3 – 4**

Instead of hand-coding the goal-state preconditions, we only specify the actions (and preconditions associated with buffers other than the goal buffer): that's the reason for striking out the goal specifications in (1)-(4). We then let the Q-learning agent learn to successfully carry out the LD tasks. We give the agent a reward of 1 if it reaches the final goal-state `done`. For any intermediate rule firing, we give it a small negative reward of -0.15 to encourage it to finish the task as soon as possible. However, the agent does not get the small penalty if it chooses to wait and fire no rule: this is the optimal course of action when waiting for retrieval requests from declarative memory to complete, for example.

The agent learns by trial and error to successfully carry out the LD tasks: it learns how to properly order the rules and complete the LD tasks as efficiently as possible. This is no small feat given that the actual number of steps, i.e., decision points, when the agents needs to select an action, is larger than the high-level sequences of rule firings discussed above. For example, for a 1-stimulus task, there are actually 12 steps where the agent needs to decide whether to wait or to fire a specific rule (when the agent does not complete the

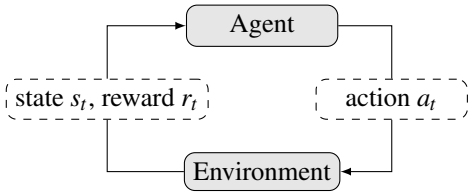


Figure 1: The agent-environment interaction in an MDP

task perfectly, it might take much more than 12 steps). The 2-stimuli task requires 18 such steps (if the task is completed perfectly), and the 4-stimuli task requires 34 steps (again, if the task is completed perfectly). The reason for this is that our LD simulations involve the visual and motor modules (to read strings of characters and press keys) in addition to the declarative memory module. Visual and motor actions, just as retrievals from declarative memory, take time, and the agent needs to make decisions while waiting for them to complete.

The higher the number of steps, i.e., the higher number of decision points for the agent, the harder the task is to learn. As the next sections show, learning is faster and less noisy for shorter tasks (fewer stimuli), but the Q-learning agent manages to learn even the most complex 4-stimuli task fairly well.

Production-rule ordering as an RL problem

Markov decision processes

Markov Decision Processes (MDPs) are the deterministic or stochastic models of decision-making sequences that form the basis of RL approaches to learning. In an MDP, an agent interacts with its environment and needs to make decisions at discrete time steps $t = 1, 2, \ldots, n$. Defining what counts as the agent and what counts as its environment is part of the modeling process: the agent could be a whole cognitive agent (animal, human or robot) acting in the world or in an experimental environment, or it could be a component of a cognitive agent interacting with an 'environment' consisting of other cognitive components.

At every time step t, all the information from the past that is relevant for the current action selection is captured in the current state of the process s_t. This is the Markov property: the future is independent of the past given the current state. The environment passes to the agent the state s_t and, at the same time, a reward signal r_t. The agent observes the current state s_t and reward r_t, and takes an action a_t, which is passed from the agent to the environment. Then, the cycle continues, as shown in Figure 1. At time step $t+1$, the environment responds to the agent's action with a new state s_{t+1} and a new reward signal r_{t+1}. Based on these, the agent selects a new action a_{t+1} etc.

The definitions of 'state' and 'action' depend on the problem, and are also part of the modeling process. The agent's *policy* is a complete specification of what action to take at any time step. Given the Markovian nature of the MPD, the policy π is effectively a mapping from the state space S to the action space A, $\pi : S \to A$. A deterministic policy is a mapping from any given state s_t to an action $a_t = \pi(s_t)$, while a stochastic policy is a mapping from any given state s_t to a probability distribution over actions $a_t \sim \pi(s_t)$.

The agent's goal is to maximize some form of cumulative reward – e.g., total reward, average reward, future-discounted sum of rewards – over an *episode*, which is a complete, usually multi-step interaction between the agent and its environment. In our case, an episode would be a full simulation of an LD task (be it a 1-stimulus or a 2/4-stimuli task).

The agent learns (solves/optimizes the MDP) by updating its policy π to maximize the per-episode cumulative reward.

The standard cumulative reward for an episodic task is the discounted return G, which is the sum of the current reward and the discounted future rewards until the final step n of the episode. In finite/episodic tasks, future rewards are discounted because we assume the agent has a preference for more immediate rewards rather than rewards in the far future. The discount factor γ is a real number between 0 and 1 that determines the present value of future rewards. The discounted return at a time step $t < n$, where n is the final step in the episode, is defined as:

(5) **Discounted return at time** t (γ is the discount factor):
$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots + \gamma^{n-t-1} r_n$

The agent selects actions with the goal of maximizing the expected discounted return.

We define the (state-)action value function $Q_\pi(s,a)$ to be the expected (discounted) return when starting in state s, performing action a, and then following the policy π until the end of the episode. With Q_π in hand, we can solve the optimization problem framed by an MDP: if we know this function, we can always select an optimal action in any state (optimal actions are actions with maximal expected return).

Q-learning, discussed in the next subsection, estimates Q_π by directly sampling experience from the environment. This is in contrast to Dynamic Programming methods, for example, which compute optimal policies under the (unrealistic) assumption that we have a perfect model of the environment.

Q-learning

The agent's goal is to maximize its return. One way of doing this is to bypass the policy and directly estimate the value of all state-action pairs, i.e., estimate the Q function, and improve this estimate based on the interactions between the agent and its environment. With a good estimate of the Q function in hand, we can devise an optimal policy by selecting a maximum-value action in each state.

Tabular Q-learning is an algorithm that enables us to estimate the Q function, and use this estimate as the basis for an optimal policy. The Q function $S \times A \to \mathbb{R}$ is represented as a look-up table that stores the estimated values of all possible state-action pairs. Before learning begins, the Q table is initialized to an arbitrary fixed value (0). The agent then updates the Q table incrementally at each time step t: the value of the pair (s_t, a_t), which consists of the state of the environment s_t

relative to which the agent took the action a_t, is updated based on the reward signal r_{t+1} and the new state s_{t+1} that the agent receives from the environment after taking action a_t.

Q-learning is a form of temporal difference (TD) learning, as the update in (6) shows. The Q^{new} value estimate for the state-action pair (s_t, a_t) is based on the Q^{old} value, updated by some proportion α of the TD error. The parameter α is the learning rate, $0 < \alpha \leq 1$. This update $\alpha \cdot$ TD error is provided on the second line of (6).

(6) **Q-learning update**:

$$Q^{new}(s_t, a_t) \quad \overbrace{Q^{old}(s_t, a_t) + \alpha \cdot \underbrace{\left(r_{t+1} + \gamma \cdot \underbrace{\max_{a_{t+1}} Q^{old}(s_{t+1}, a_{t+1})}_{\text{next-state value estimate}} - Q^{old}(s_t, a_t) \right)}_{\text{TD target (updated value)}}}^{\text{TD (temporal difference) error}}$$

The TD error is the difference between the TD target – which is an updated estimate of the value of the (s_t, a_t) pair – and the Q^{old} value estimate. The TD target consists of:

- the reward r_{t+1} the agent receives after action a_t, which is part of the new data the agent gets back from the environment after action a_t, plus
- the estimate of the value of the next state s_{t+1}, where the next state s_{t+1} is the other part of the new data the agent gets back from the environment after action a_t.

The Q-learning estimate for the value of the next state s_{t+1} is discounted by the discount factor γ, since this state is in the future relative to the state-action pair (s_t, a_t) we're currently updating. The Q-learning value estimate for s_{t+1} is aggressively confident/optimistic:[1] the agent looks at all the possible actions a_{t+1} that can be taken in state s_{t+1} and confidently assumes that the action a_{t+1} with the highest Q^{old}-value provides an accurate estimate of the s_{t+1} value.

Q-learning for production-rule ordering

Returning to our ACT-R model of LD tasks, *the agent* (in the RL sense) is a Q-value table that assigns values to all possible state-action pairs, and that guides the rule selection process at every cognitive step. *The environment* is the cognitive state of the ACT-R model/mind, and could conceivably consist of:

i. all the modules – procedural memory, declarative memory and visual and motor modules, together with
ii. their associated buffers – goal, retrieval, visual-what, visual-where and the manual buffer.

This, however, would lead to a very large state space S, which in turn would lead to a large Q-value table. Function-approximation approaches – e.g., using neural networks – would mitigate this problem, but we will continue using a tabular approach here and take a state s to only consist of:

i. the current goal buffer,
ii. the current retrieval buffer,
iii. the value in the visual-what buffer, if any (otherwise, we explicitly mark the visual-what buffer as having no value), and finally,
iv. the state of the manual buffer (busy or free).

Four example states are provided below:

```
- {goal: {state: retrieving}, manual: free,
   retrieval: {}, visual_value: NO_VALUE}
- {goal: {state: retrieval_done}, manual: free,
   retrieval: {}, visual_value: elephant}
- {goal: {state: retrieval_done}, manual: free,
   retrieval: {form: elephant},
   visual_value: elephant}
- {goal: {state: done}, manual: busy,
   retrieval: {}, visual_value: NO_VALUE}
```

The action space consists of the 4 rules in (1)-(4) above, namely `retrieving`, `lexeme retrieved`, `no lexeme found` and `finished`, together with a special action `None` that the agent selects when it wants to not fire any rule because it prefers to wait for a new cognitive state.

The full details of the reward structure are as follows:

- the agent receives a positive reward of 1 at the end of an episode (when the LD task is completed), specifically, when the goal STATE is `done`;
- the agent receives a negative reward of -0.15 for every rule it selects (other than `None`);
- there is no penalty for waiting and selecting no rule, i.e., for selecting the special action `None`;
- at every step, the agent receives a negative reward equal to the amount of time that has elapsed between the immediately preceding step and the current step (multiplied by -1 to make it negative).

This reward structure is designed to encourage the agent to finish the task as soon as possible, and in the process select the smallest number of rules. The negative temporal reward in particular discourages the agent from just repeatedly selecting the special action `None`. This would end up timing out the LD task in a small number of steps, and it would fast-forward the agent to the maximum waiting time the ACT-R environment allows for, which we set to 2 seconds per word for LD tasks.

Thus, we work with two time 'counters' here. On one hand, we have the continuous cognitive-process time that the ACT-R model/mind keeps track of, and which models the reaction time of human participants in experimental tasks. On the other hand, we have the discrete RL time that is the counter for agent-environment interactions: the discrete time steps $t = 1, 2, \ldots, n$ in our MDP. From the perspective of the discrete RL/MDP time, the continuous ACT-R time is a feature of the environment, reflected in the reward signal that the (Q-learning) agent gets when it interacts with, i.e., samples experience from, the environment.

[1] In contrast to the Expected Sarsa estimate, for example (van Seijen, van Hasselt, Whiteson, & Wiering, 2009).

Simulations and results

We assume the usual ACT-R defaults, e.g., rule firing time is set to 50 ms. The learning rate α is set to 10^{-3}, and γ to 0.95. We use an ε-greedy policy for all simulations, with ε multiplicatively annealed from a starting value of 1 to a minimum value of 0.01. Specifically, at every RL step (after every action/rule selection), if $\varepsilon > 0.01$, its value is updated as follows: $\varepsilon \leftarrow \varepsilon \cdot (1 - 10^{-5})$.

One-stimulus task

We simulate 15,000 episodes, i.e., 15,000 LD tasks consisting of 1 stimulus only (the word 'elephant'), from which the *Q*-learning agent learns. The plot in Figure 2 shows that, after about 5,000 episodes, the task is completed in ≈ 12 steps, which is the length of the task when the agent completes it perfectly. For some episodes, the number of steps is smaller than 12. In these cases, the agent times out the task, e.g., by selecting the `None` action several times, and receives steeply negative temporal rewards leading to very low returns.

A close examination of the agent's final *Q*-value table, which stores the agent's rule-firing preferences for any given goal state, indicates that the agent has learned goal-conditioned rules perfectly. We only look at states for which at least one action/rule has a non-0 value (recall that all *Q*-values are initialized to 0). For each such state, we identify the action/rule with the highest value. There are 8 states with at least one non-0 value action. Let's examine them.

There are 3 states in which the agent fires no rule, that is, the maximum-value action is `None`:

```
- {goal: {state: retrieving}, manual: free,
   retrieval: {}, visual_value: NO_VALUE}
- {goal: {state: retrieving}, manual: busy,
   retrieval: {}, visual_value: NO_VALUE}
- {goal: {state: retrieval_done}, manual: free,
   retrieval: {}, visual_value: elephant}
```

We see that when the goal state is `retrieving`, the retrieval buffer is empty, and the visual buffer stores no value, the agent does nothing (whether the manual buffer is busy or free): it simply waits for some text to be automatically detected and stored in the visual buffer. Similarly, when the goal state is `retrieval_done`, the visual buffer stores the value `elephant`, but the retrieval buffer is empty, the agent once again does nothing: it waits for the retrieval process that was just started to complete.

There are 3 states where the max-value action is `finished`:

```
- {goal: {state: retrieving}, manual: busy,
   retrieval: {}, visual_value: FINISHED}
- {goal: {state: retrieval_done}, manual: free,
   retrieval: {form: elephant},
   visual_value: FINISHED}
- {goal: {state: retrieving}, manual: free,
   retrieval: {}, visual_value: FINISHED}
```

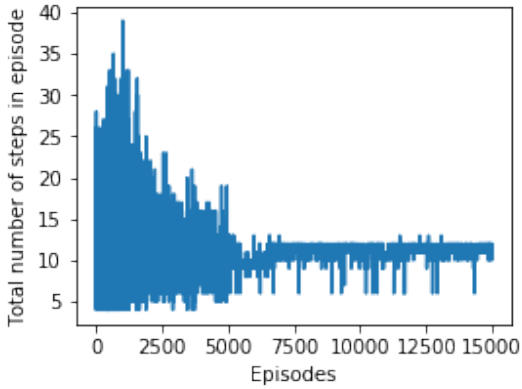


Figure 2: Steps per simulation to complete the 1-stimulus task

We see that whenever the value stored in the visual buffer is FINISHED, the agent correctly chooses the `finished` action irrespective of: (i) the goal state, (ii) the state of the manual buffer, or (iii) the contents of the retrieval buffer.

This leaves us with the 2 state-action pairs below:

```
- {goal: {state: retrieving}, manual: free,
   retrieval: {}, visual_value: elephant}
   ==> retrieving
- {goal: {state: retrieval_done}, manual: free,
   retrieval: {form: elephant},
   visual_value: elephant}
   ==> lexeme retrieved
```

We see that when the goal state is `retrieving` and the visual value is `elephant`, the agent correctly chooses the `retrieving` rule. Finally, when the goal state is `retrieval_done` and the retrieval buffer contains the word `elephant` (i.e., the retrieval process has been successful), the agent correctly chooses the `lexeme retrieved` rule.

Thus, there is no need to hand-code goal states in the conditions of a rule to deterministically guide the cognitive process – at least for this simple 1-stimulus LD task. The *Q*-learning agent learns by trial-and-error interaction with the environment when to fire which rule, and when to choose to wait and not fire any rule. The agent learns all this from a minimal, but fairly carefully designed, reward structure.

Two-stimuli task

We simulate 15,000 episodes, i.e., 15,000 LD tasks consisting of 2 stimuli only (the word 'elephant' and the non-word 'not_a_word'), from which the *Q*-learning agent learns. The plot in Figure 3 shows that, after about 9,000 episodes, the task is completed in ≈ 18 steps, which is the length of this task when the agent completes it perfectly.

A close examination of the agent's final *Q*-value table indicates that the agent has learned goal-conditioned rules almost perfectly. Once again, we only look at states for which at least one action has a non-0 value – a total of 13 states. For each state, we identify the maximum-value action.

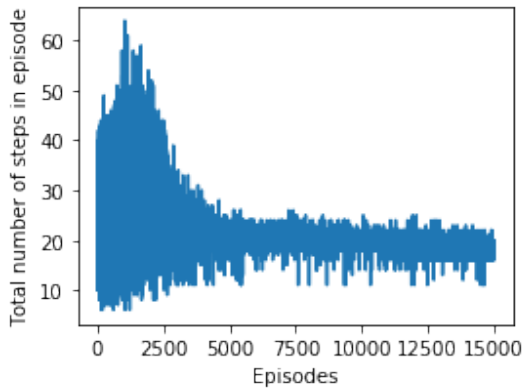


Figure 3: Steps per simulation to complete the 2-stimuli task

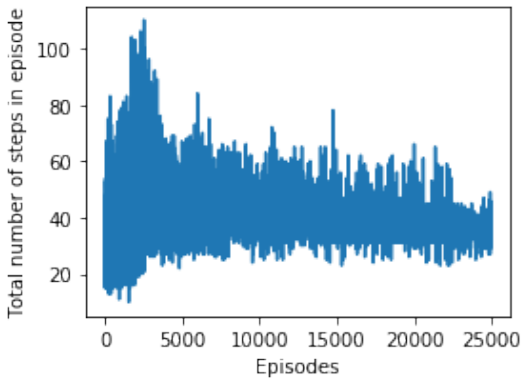


Figure 4: Steps per simulation to complete the 4-stimuli task

There are 4 states where the agent does nothing (selects the `None` action): while waiting for the word 'elephant' to be retrieved, while waiting for a visual value to be automatically detected and stored in the visual buffer (whether the manual buffer is busy or free), and while waiting for the retrieval of the text 'not_a_word' to fail.

There are 4 states where the agent correctly fires the `finished` rule: the visual value is FINISHED in all of them.

There are 5 state-action pairs in which the action is one of the other 3 rules. Out of these, 4 state-action pairs are exactly what we would expect:

- when the goal state is `retrieving`, the retrieval buffer is empty and the manual buffer is free, whether the visual value is `elephant` or `not_a_word`, the agent correctly fires the `retrieving` rule;
- when the goal state is `retrieval_done` and the retrieval process has completed successfully based on the visual value `elephant`, the agent correctly fires the `lexeme retrieved` rule;
- finally, when the goal state is `retrieval_done` and the retrieval process has completed unsuccessfully based on the visual value `not_a_word`, the agent correctly fires the `no lexeme found` rule.

However, unlike in the 1-stimulus task, there is 1 state-action pair that is not optimal, and is simply a reflection of the trial-and-error learning process that takes longer is and more error prone than for the simpler, 1-stimulus task. This state-action pair is the following:

- the goal state is `retrieval_done`, the retrieval buffer contains the word `elephant`, but the visual value is `not_a_word`; in such a state, the agent fires the `retrieving` rule, which is the maximum-value action.

Four-stimuli task

We simulate 25,000 episodes, i.e., 25,000 LD tasks consisting of 4 stimuli (the word 'elephant', the non-word 'not_a_word', the word 'dog' and the non-word 'not_a_word_again'), from which the Q-learning agent learns. We need more episodes for this task because it is longer, hence more complex, than the 1-stimulus or the 2-stimuli tasks. The plot in Figure 4 shows that it takes about 22,000 episodes for the task to be reliably completed in less than 40 steps. The task takes 34 steps when the agent completes it perfectly.

A close examination of the agent's final Q-value table indicates that the agent has learned goal-conditioned rules fairly well, but there still is a pretty large amount of noise associated with multiple goal states. Specifically, there are 24 states total with at least one action with a non-0 value. When we examine the maximum-value action for each of these states, 18 state-action pairs make sense and are as expected.

However, the remaining 6 state-action pairs do not make much sense, similar to the final state-action pair we discussed in the 2-stimuli task subsection. This noise is a reflection of the trial-and-error learning process that becomes increasingly difficult for tasks requiring large numbers of steps. With 4-stimuli, we see that even after 25,000 episodes, the agent still wastes time every now and then trying incorrect rules, or just waiting (selecting the action `None`) for no good reason.

Conclusion and future directions

We argued that the learnability problem of production-based cognitive models can be systematically formulated and computationally addressed as a reinforcement learning problem. But this is merely a first inroad into what promises to be a very rich nexus of learnability questions.

For example, an immediate follow-up would be to explore how RL algorithms perform on a variety of production-based cognitive models, whether linguistic, e.g., syntactic or semantic parsing, or non-linguistic. We have conducted pilot experiments with simple parsing models and tasks, and they are much more difficult than the LD tasks explored in this paper.

One reason is that the cardinalities of the state and action/rule spaces are much larger than for LD tasks, which makes tabular learning less effective and prompts us to explore function-approximation approaches, e.g., neural-

network based approaches (Deep Reinforcement Learning, see Mnih and al 2015 among many others).

The second reason is that parsing tasks are much longer – many more RL/MDP time steps, i.e., action-selection decision points – and standard RL methods are increasingly ineffective in this kind of sparse-reward, long time horizon tasks because the temporal credit assignment problem becomes very difficult. The difficulty is further increased when function-approximation approaches are used – because of vanishing-gradient issues among others. It would be worth exploring function-approximation approaches in realistically long LD tasks first (hundreds of LD stimuli, hence many steps per episode), and only subsequently explore parsing tasks.

This leads us to a second cluster of learnability issues that could be explored. There are other value-based tabular learning algorithms (Sarsa, Expected Sarsa), as well as non-tabular approaches to reinforcement learning (both value and policy based), e.g., linear or non-linear (neural network) function-approximation approaches. We already indicated that exploring these algorithms will most probably be necessary for more complex tasks like parsing, but it is worth exploring their performance and comparing them to Q-learning on LD decision tasks first (both on the tasks we used here, and on realistically long tasks with hundreds of stimuli), so that we establish a broad set of baselines before moving on to other linguistic and non-linguistic production-based cognitive models.

Similarly, we might want to investigate curriculum learning (see Elman 1993 for an early reference, and Rusu and al 2016 among many others for a more recent discussion) for increasingly complex tasks. For example, we could design a curriculum that starts with LD tasks with fewer stimuli and scales up to realistically long LD tasks. Similarly, we could start with parsing short sentences and scale up to sentences with a variety of embedded clauses, or even multi-sentential discourses, after which we could scale up to realistically long self-paced reading or eye-tracking tasks with tens or hundreds of such multi-clausal sentences or discourses.

Curriculum or transfer learning should also enable us to address the fact that it is not cognitively realistic to require such a high number of episodes/trial-and-error interactions for learning. The human cognitive architecture enables us to learn from much fewer interactions, and/or from explicit instructions. For tabular Q-learning, for example, this would mean that the agent starts with a pretrained Q-table.

Finally, a separate line of future work would go beyond our current focus on the easier problem of rule ordering, and investigate the extent to which Hierarchical RL methods (Sutton, Precup, and Singh 1998 among others) could be brought to bear on the harder problem of rule acquisition.

Acknowledgments

We are very grateful to two anonymous ICCM 2020 reviewers for their detailed feedback on an earlier version of this paper, and to the audience of the UCSC Linguistics Department S-circle (May 2020) for their questions and comments about this research project. We gratefully acknowledge the support of the NVIDIA Corporation with the donation of two Titan V GPUs used for this research, as well as the UCSC Office of Research and The Humanities Institute for a matching grant to purchase additional hardware.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Brasoveanu, A., & Dotlačil, J. (2019). Quantitative comparison for generative theories. In *Proceedings of the 2018 berkeley linguistic society 44*.

Brasoveanu, A., & Dotlačil, J. (2020). *Computational Cognitive Modeling and Linguistic Theory*. Springer (Open Access). doi: https://doi.org/10.1007/978-3-030-31846-8

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, *48*, 71-99. doi: 10.1016/0010-0277(93)90058-4

Engelmann, F. (2016). *Toward an integrated model of sentence processing in reading*. Unpublished doctoral dissertation, University of Potsdam, Potsdam.

Fu, W.-T., & Anderson, J. R. (2006). From recurrent choice to skill learning: A reinforcement-learning model. *Journal of Experimental Psychology: General*, *135*(2), 184-206. doi: 10.1037/0096-3445.135.2.184

Hale, J. (2011). What a rational parser would do. *Cognitive Science*, *35*, 399–443.

Lewis, R., & Vasishth, S. (2005). An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, *29*, 1-45.

Mnih, V., & al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Rusu, A. A., & al. (2016). *Progressive neural networks.*

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., Precup, D., & Singh, S. P. (1998). Intra-option learning about temporally abstract actions. In *ICML 1998* (pp. 556–564). Morgan Kaufmann.

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say "broke"? a model of learning the past tense without feedback. *Cognition*, *86*(2), 123–155.

van Seijen, H., van Hasselt, H., Whiteson, S., & Wiering, M. (2009). A theoretical and empirical analysis of Expected Sarsa. In *Ieee symposium on adaptive dp and rl* (p. 177-184).

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Unpublished doctoral dissertation, King's College, Cambridge, UK. Retrieved from http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292. Retrieved from https://doi.org/10.1007/BF00992698 doi: 10.1007/BF00992698

A Distributed Spiking Neuron Model of Attention in the Stroop Task

Emilie Caron (emilie.caron@uwaterloo.ca)
Department of Psychology, University of Waterloo
200 University Avenue West, Waterloo, ON, Canada, N2L 3G1

Terrence C. Stewart (terrence.stewart@nrc-cnrc.gc.ca)
National Research Council of Canada, University of Waterloo Collaboration Centre
200 University Avenue West, Waterloo, ON, Canada, N2L 3G1

Abstract

We present a spiking neuron-based model of the Stroop task where the attention mechanism is entirely implemented with distributed representations. This is done by using the Neural Engineering Framework and the associated Semantic Pointer Architecture to implement a selective attention mechanism. The resulting system exhibits the Stroop effect, as well as the associated Facilitation and Interference effects. In contrast with previous models, these effects are not generated via a localist competition mechanism. Rather, these effects are a result of controlled unbinding of information from a combined distributed representation.

Keywords: Stroop task; neural engineering framework; semantic pointer architecture; spiking neurons; distributed representation

Introduction

Current models of the neural mechanisms underlying selective attention (in tasks such as the Stroop task) rely on a localist representation of concepts. That is, they postulate that there are individual and separate neurons (or groups of neurons) representing concepts such as RED and BLUE, and whether attention should be paid to the COLOR of a word or to the WORD itself. While these sorts of localist neural models are common, new methods have been developed for creating neural models that make use of distributed representations, where the color BLUE would be represented by a particular pattern of activity over a group of neurons, and a different pattern of activity would represent RED (or a concept such as COLOR or WORD, etc.). These sorts of representations match well to biology (e.g. Stewart & Eliasmith, 2012) and offer an alternative set of mechanisms for the manipulation and control of representations. Here, we apply these techniques for the first time to modelling the Stroop task.

Background

On a daily basis, individuals are tasked with allocating their attention to specific information given their situational demands. This is done by selectively choosing to focus on the relevant aspects of their situation and discarding the irrelevant ones (Bustamante, Lieder, Musslick, Shenhay, & Cohen, 2020). This behaviour is typically believed to be guided by our internal state and often explained using the Top-down Excitatory Biasing (TEB) model. Specifically, the TEB model suggests that representations of cognitive control guide task completion through heightened levels of activity in groups of neurons associated with processing task-relevant information in relation to the levels of activity in groups of neurons associated with processing task-irrelevant information (Corbetta, Miezin, Dobmeyer, Shulman, & Petersen, 1991; Herd, Banich, & O'reilly, 2006). As a result, irrelevant information becomes less influential.

A common task in which this phenomenon is illustrated is the Stroop task (Stroop, 1935). The Stroop task is often referred to as a control task or a conflict-related task (Petersen & Posner, 2012) because individuals are required to selectively attend and respond to the pertinent information from the stimulus and ignore the impertinent information. Specifically, when completing the Stroop task, participants are instructed to respond as quickly and accurately as possible by naming the color of a word stimulus, all while ignoring the occasionally-distracting information coming from the word itself. Accordingly, the Stroop task typically consists of two kinds of stimuli: 1) congruent stimuli, where the word is a color word that matches the color in which it is presented (e.g., the color word `RED` presented in red) and 2) incongruent stimuli, where the color word mismatches the color in which it is presented (e.g., the color word `RED` presented in blue). Importantly, individuals typically take longer to name the color of the stimuli on the incongruent trials than on congruent trials. This difference in response time between congruent and incongruent word stimuli is often referred to as the Stroop effect and is commonly perceived to be a result of the automaticity of word reading which influences color naming on incongruent and congruent trials (e.g., Ashcraft, 1994; Rayner & Pollatsek, 1989). That is, when responding to congruent trials participants might be enlisting well-established reading processes to enhance the speed and accuracy of their responses, whereas on the incongruent trials participants might need to depend on cognitive control systems to reduce the reflex to read the color word and initiate the color-naming processes (Bugg, Jacoby, & Toth, 2008).

Notably, current neural models of the Stroop task (Botvinick, Braver, Barch, Carter, & Cohen, 2001; Cohen, Dunbar, & McClelland, 1990; Cohen & Huston, 1994; Herd et al., 2006) have demonstrated this conflict-control mechanism whereby the representation of the activity in the simulated neurons differs when responding to the congruent trials versus responding to the incongruent trials. Thus, the representation of neural activity is heightened in response to congruent trials, where naming the color of the color word stim-

ulus is assumed to be supported by the automatic process of reading the word. At the same time, the representation of neural activity is weakened when responding to the incongruent trials, where naming the color of the color word stimulus is assumed to be hindered by the automatic process of reading the word (e.g., Herd et al., 2006; Bugg et al., 2008).

Critically, a common trait shared by these current models is their dependence on the notion of a localist representation. In other words, there are separate representations for each individual concept, such as RED or BLUE. These separate representations compete with each other, in that as the representation of RED becomes stronger, it reduces or inhibits the representation of BLUE (and all other competing terms). For models that make use of neural mechanisms, these separate representations often take the form of separate neurons, or separate groups of neurons, one group for each possible concept. These localist representation models are a relatively common method of examining conflict-related tasks such as the Stroop task. However, this sort of localist representation has been criticized and contrasted with distributed representations (e.g. Rumelhart, McClelland, & PDP Research Group, 1986), and these types of distributed representations can form closer connections to the underlying neurobiological implementation (e.g. Stewart, Bekolay, & Eliasmith, 2011).

With this in mind, we aim to create a novel model of the Stroop task that makes use of distributed representations. We believe this framework may provide more information regarding the function of the neural system on cognitive control. In addition to the use of distributed representations in our model, we also include a third kind of stimuli, the neutral word stimuli. The Neutral stimuli is all-too-frequently excluded from the Stroop task, despite it being useful for interpreting how much of the Stroop effect is driven by Interference (arising from the incongruent words) and how much of it is driven by Facilitation (arising from congruent words) (MacLeod, 1991). Although there is debate as to what consists of an appropriate neutral stimulus, for the sake of our paper we have opted to use non-color word stimuli which is a commonly used method in the color Stroop task.

The inclusion of the neutral stimuli allows us to break down the Stroop effect into its various parts and to explore these complexities in greater detail. Further, by implementing distributed representations within the model we are suggesting a very different mechanistic theory than other models of the Stroop Task. In particular, rather than relying on competitive inhibition operations, our model is based on computing mathematical operations on high-dimensional vectors. Since these operations can then be implemented using biologically detailed neurons, we can tie the model more closely to biological constraints. This allows for direct output of metrics such as response times, rather than using an abstract notion of time steps.

The goal of our paper is to demonstrate a neural mechanism that can produce the Stroop effect so that we might be able to evaluate in greater depth what is occurring at the neural level. By doing so, we hope to show exactly how top-down biasing can be implemented in a flexible manner. We also hope it can shed light on why the magnitude of the Stroop effect varies based on different situations, (e.g. 160-260ms in Dunbar & MacLeod, 1984 and 75-150ms in Augustinova, Parris, & Ferrand, 2019). Furthermore, we examine the relationship between the Stroop effect and its two components: the Facilitation effect (defined as the difference in performance between the congruent and neutral trials, i.e. the benefit when the word is congruent to its color) and the Interference effect (defined as the difference in performance between the incongruent and neutral trials; i.e. the penalty when the word is incongruent to its color).

Distributed Representation

Most models of language that use distributed representations make use of vectors (i.e. a list of numbers). That is, each basic term (RED, BLUE, COLOR, etc.) is a vector in some high-dimensional space. These vectors can be randomly chosen, or can be chosen to respect semantic similarity (so that the vector for RED is similar to the vector for PINK, for example). In the work presented here, all vectors are randomly chosen 512-dimensional unit-length vectors.

Importantly, these distributed representations can be combined in order to create representations of more complex structures. While there are many different mathematical frameworks suitable for forming these combinations (see Levy & Gayler, 2008 for an overview), they all generally have operators for binding and unbinding. For example, one can build a vector to represent "dogs chase cats" by computing the following:

$$\mathtt{S} = \mathtt{SUBJECT} \circledast \mathtt{DOGS} + \mathtt{VERB} \circledast \mathtt{CHASE} + \mathtt{OBJECT} \circledast \mathtt{CATS}$$

This gives us a final resulting vector S which forms a distributed representation of that entire sentence. Given S, we can recover the individual parts by performing unbinding (written as binding by the inverse, noted by $^{-1}$). In this case, if we want the object of the sentence, we can compute

$$\mathtt{S} \circledast \mathtt{OBJECT}^{-1} \approx \mathtt{CATS}$$

This is, of course, an approximation, and the resulting output will be less and less similar to the ideal vector for CATS as the vocabulary size and the number of terms that are combined (and the depth of the combinations) are increased.

In this work, we use circular convolution as a binding operation, and its associated pseudo-inverse (turning circular correlation into circular convolution) as the unbinding operator, as suggested in (Plate, 1995). These operations have been shown to be efficiently implementable by spiking neurons, and have been used in many neurally detailed models of cognitive behaviour, including sequential memory (Choo & Eliasmith, 2010), semantic search (Kajić, Gosmann, Stewart, Wennekers, & Eliasmith, 2017), emotional appraisal (Kajić, Schröder, Stewart, & Thagard, 2019), and spatial representation (Lu, Voelker, Komer, & Eliasmith, 2019). This approach is known as the Semantic Pointer Architecture, and has been

shown to scale efficiently to human-sized vocabularies and sentence structures (Eliasmith, 2013).

Distributed Representation of the Stroop Task

While distributed representations and Semantic Pointers have been used to model a variety of tasks, they have not yet, to our knowledge, been used to model the Stroop task. To do this, we take an approach that closely follows the binding and unbinding ideas described in the previous section.

First, we are not modelling the entire visual processing system. Instead, we assume that visual processing is inputting to our model a combined representation of the word and its color (and any other visual information could also be included). That is, the input representation is a vector that combines the vector for the word with the vector for the color. For example, the word RED written in blue would be presented to the model as:

$$\texttt{vision} = \texttt{WORD} \circledast \texttt{RED} + \texttt{COLOR} \circledast \texttt{BLUE}$$

This can be thought of as the output (final layer) of a standard deep-network vision system which is external to the model presented here, and in future work we will be including this vision system as part of the model.

In the normal Stroop task, participants are asked to identify the color of the stimuli they are seeing, but it is also possible to ask them to identify the words. This means there must be a way to dynamically change the aspect of the visual stimuli to which participants are paying attention. In our model, this is a separate input. If they are perfectly paying attention to the color, then the vector input to this part of the model is the vector COLOR.

This approach allows us to explore one possible method for producing the Stroop effect: attention might not be perfect. Instead, the attention value might be $\texttt{attention} = \texttt{0.7COLOR+0.3WORD}$, indicating that most of the attention is on the color of the word, but some attention is on the word itself.

Given this setup, the model itself just needs to compute the result of $\texttt{vision} \circledast \texttt{attention}^{-1}$. This results in a vector that should be close to the concept that determines the response.

However, in order to produce a response time out of our model, we also need to include a mechanism that will take the resulting vector and turn it into an explicit response (saying a color name, or pressing a button). To model this, we use a set of independent accumulators (Gosmann, Voelker, & Eliasmith, 2017). That is, we take the result value and feed it into a set of evidence accumulators, one for each possible response. This means there is a group of neurons whose activity increases based on similarity between the result and the vector for RED, another for BLUE, and so on. (Similarity is computed as the dot product). When this activity reaches a threshold, we consider that to be the time that a response is generated. Making a response suppresses activity in all the other accumulators, but there is no interaction between the accumulators until that decision is generated.

It should be noted that the only part of this model which uses localist representations is the independent accumulators used to make the final decision. All other parts of the model are distributed, and we do not make use of any direct competition mechanism, unlike the localist models discussed above.

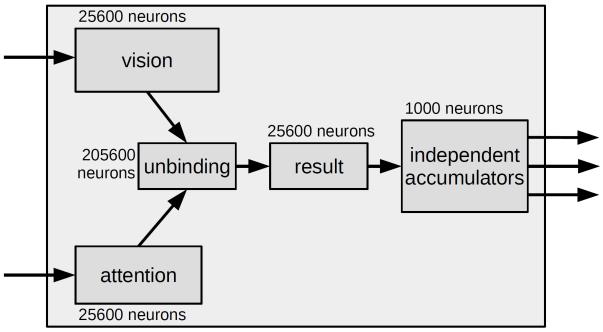


Figure 1: The model of attention in the Stroop task. Inputs are vectors, such as $\texttt{WORD} \circledast \texttt{RED} + \texttt{COLOR} \circledast \texttt{BLUE}$ for the vision and $\texttt{0.7COLOR+0.3WORD}$ for the attention. Boxes represent single layers of neurons and arrows represent all-to-all connection weight matrices. Connection weights are set to optimally approximate the desired unbinding operation (circular correlation), and the independent accumulators are a set of neural integrators that build up evidence over time until reaching a threshold and producing an output response.

Neural Implementation

Previous work has shown how spiking neurons can be connected such that they represent vectors and compute functions such as circular convolution (Eliasmith, 2013), and we follow the same approach here. To have a group of neurons represent a vector, we randomly assign each neuron a "preferred direction vector". This is the vector for which it will fire most strongly, consistent with the preferred stimuli found for many sensory and motor neurons (e.g. Georgopoulos, Schwartz, & Kettner, 1986). Each neuron is also given a random gain and bias current, providing heterogeneity in the population coding. In this situation, any vector will result in a different pattern of neural activity in the population, and we can think of this as much like a single-hidden-layer neural network where the input weights are randomly chosen.

In order to compute functions using these distributed representations, we solve for the connection weights that will lead to one group of neurons causing the desired neural activity in another group of neurons. For example, if one group of neurons represents x and another group of neurons represents y and we want $y = f(x)$, then we need to find the set of connection weights from the x population to the y population that achieves this for all x values. This can be solved using a variety of optimization techniques; here we treat it as a least-squares minimization problem and solve for the optimal connection weights. This general process is known as

the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003).

This resulting network is depicted in Fig. 1, and its behaviour is in Fig 2. The neural activity (i.e. individual neuron spikes) of a subset of the neurons in each area are shown. For the two input areas (`visual` and `attention`), we also show the computed vector that is being used to stimulate those neurons. All other neural activity is purely the result of the computed connection weights that implement the unbinding and accumulation operations. For the accumulator neurons, we also present the aggregate activity (red and blue lines) encoding the gradual increase in evidence for a decision, and the final decision that is made once a threshold is reached. The bottom row of Fig 2 shows a response time of 340ms for the incongruent case, while the congruent case is faster at 240ms.

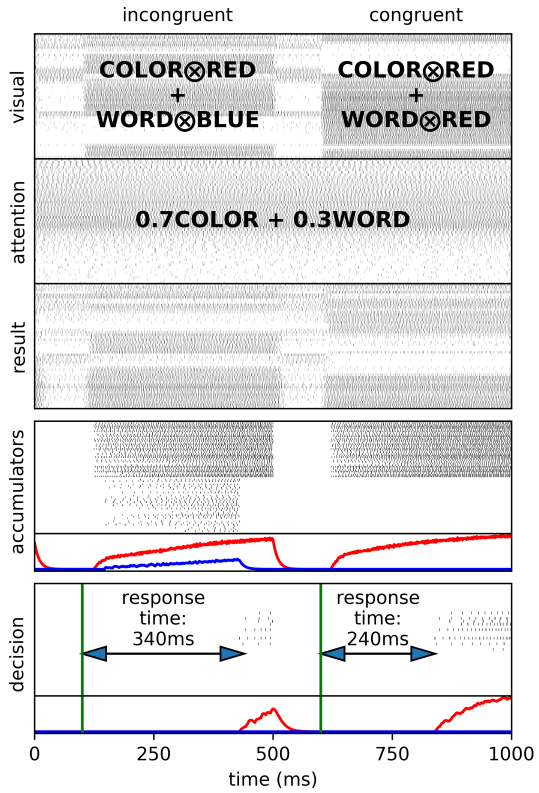


Figure 2: Model behaviour for a congruent and an incongruent example. Grey background is spiking neural activity from randomly chosen neurons. Overlaid text is the vector presented as input. Red and Blue lines show the accumulation of the similarity between the neural activity in `result` and the ideal neural activity for `RED` and `BLUE`, respectively. The decision is made when an accumulator reaches a threshold, which then suppresses the other accumulators.

Results

In this paper, we examine the overall behaviour of our model, rather than fitting it to a particular study. The magnitude of Stroop effects have been shown to be sensitive to a wide variety of factors (e.g. Dunbar & MacLeod, 1984; Augustinova et al., 2019). In future work we will be examining these factors and mapping them into our model, but here we just show the basic exploratory results. In general, most Stroop studies find the fastest response times for congruent trials (e.g. the word RED written in the color RED), and the slowest response times for incongruent trials (e.g. the word BLUE written in the color RED), with neutral trials somewhere inbetween (e.g. the word HOUSE written in the color RED). This basic pattern is replicated by our model (Fig. 3).

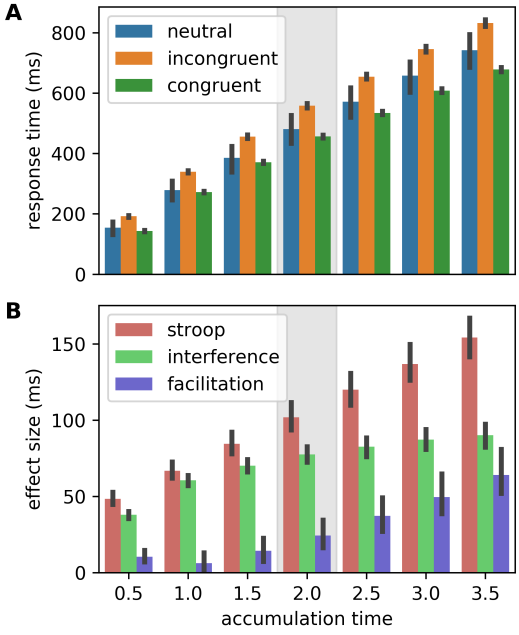


Figure 3: Model response times when varying the time over which the accumulators build up evidence. (A): Raw response times for neutral, incongruent, and congruent cases. (B): Effect sizes computed as differences in response times. Stroop = Congruent - Incongruent; Interference = Incongruent - Neutral; Facilitation = Neutral - Congruent. Shaded area shows the parameter setting used in the rest of the paper.

The actual magnitude of the response times is strongly dependent on the rate of accumulation used to make the final decision. Fig. 3A shows this linear increase in overall response time as we increase the accumulation time parameter for the independent accumulators. However, if we present the same data in a different way, Fig. 3B shows a surprising result. Here, we measure the Stroop effect (the difference in response times between congruent and incongruent trials), the Interference effect (the difference in response times between neutral and incongruent trials), and the Facilitation effect (the difference in response times between congruent and neutral trials). The Stroop effect will always be the sum of the Interference and Facilitation effects. Fig. 3B shows that increasing the accumulation time increases the Stroop effect linearly, but

the Interference effect stops increasing at around 80ms, and after that the Facilitation effect is primarily responsible for the increase in the Stroop effect. Another way to think of this is that in situations with fast response times (i.e. low amounts of accumulation time), the Stroop effect is almost entirely driven by the Interference effect, but with longer response times the Facilitation effect becomes stronger.

While we are still conducting a literature review as to how the Stroop, Interference, and Facilitation effects vary across experimental conditions, so that we can perform parameter fitting to those conditions, it should be noted that our recent experimental work has shown Interference effects in the 100-140ms range, and Facilitation effects in the 0-10ms range, for conditions where subjects are standing or sitting (Caron et al., in press).

In addition to the accumulation time, we also explored one other parameter of our basic model. The neural representation for the attention signal was initially set to `0.7COLOR+0.3WORD`. In Fig. 4 we vary this ratio. Interestingly, while this also linearly increases the Stroop effect, it causes *no change at all* to the Interference effect. Instead, this purely changes the Facilitation effect. Furthermore, if this ratio is made to be too extreme (i.e. if the attention signal becomes `0.8COLOR+0.2WORD`), then the model starts producing a *negative* Facilitation effect (i.e. congruent trials become slower than neutral trials). This is not a phenomenon that is commonly seen in the behavioural literature, which either indicates a lower bound on this parameter in humans (i.e. humans don't adjust their attention to be that extremely focused on color), or an indication that other features need to be added to the model.

As an attempt to add another feature to the model that might increase the Facilitation effect (and stop it from becoming negative), we also tried adding a direct connection between the `visual` neurons and the `result` neurons, bypassing the unbinding system. Biologically, these would be connections from the visual system that cannot be modulated by the attention system. This can be thought of as a sort of "automaticity", in that these connections are always feeding the `WORD` information to the result, no matter where high-level cognition is directing attention. The result of adding this parameter is shown in Fig 5.

With this parameter, the Interference effect is mostly unchanged, and it caused the desired increase in the Facilitation effect. However, after a certain point, it starts causing a strong drop in the Interference effect. Oddly, this does not correspond to any decrease in accuracy, and we are still analyzing the system to determine why this may be happening.

All other parameters in the model were left at the default values that have been used in previous Nengo and SPA models. Future work will analyze the effects of these other parameters, such as the number of neurons and the vector dimensionality.

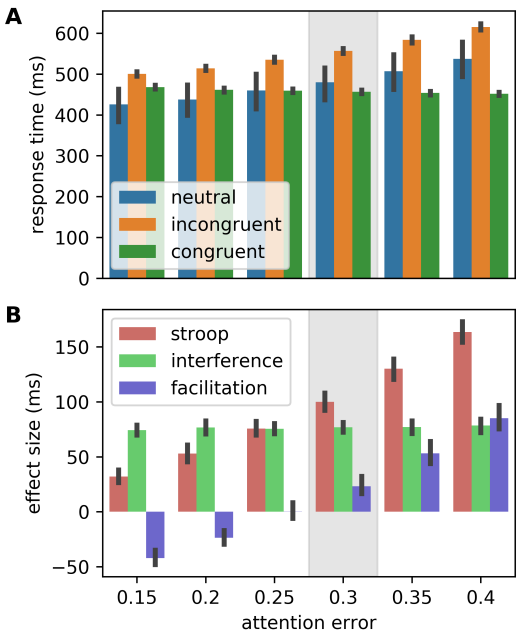


Figure 4: Model response times when varying the accuracy of the attention representation. For an attention error of x, the neural activity in the attention neurons is set to represent the vector $(1-x)$`COLOR+`x`WORD`.

Discussion and Future Work

We have presented a neural model of the Stroop task that uses a very different mechanism than previous models. Rather than relying on a localist representation and a competition-based mechanism to exhibit the Stroop effect, we produce the Stroop effect by forming distributed representations that bind together color and word information, and implement an attention computation that extracts out the desired information. As we show, this also produces a Stroop effect. We show that this can all be implemented in spiking neurons using the same mechanisms that have been instrumental in implementing other tasks, such as sequential memory, semantic search, emotional appraisal, and spatial relations.

However, we have only begun to analyze this model and cannot yet claim that it is an improvement over previous approaches. The actual magnitude of Stroop effects varies significantly in different conditions, and we have not yet begun to map different conditions into different parameter settings for our model. Our initial parameter exploration indicates that the magnitude of the Interference effect is, surprisingly, generally unaffected by the amount of attentional error or direct automaticity, and is instead only affected by the rate of accumulation of evidence needed to make a decision, and it seems to have a soft maximum of just under 100ms.

While our primary future work is to explore the parameter space of this model and fit it to various Stroop conditions, there are also clearly many additions still needed to the

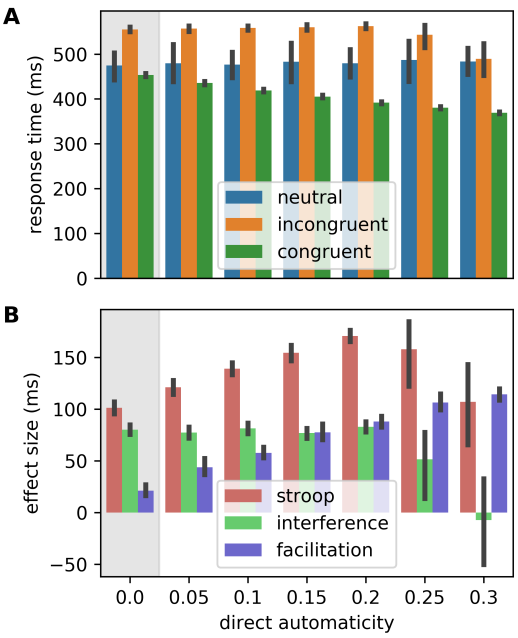


Figure 5: Model response times when introducing a direct connection between `visual` and `result` that always decodes the `WORD`, bypassing the attention system.

model. This includes adding a more detailed visual system, potentially modelling visual effects such as how increasing the spacing between letters in a word can reduce the Stroop effect. Furthermore, non-neural models of the Stroop effect (e.g. Lovett, 2005) have included cognitive strategies, which can be added to our model using the existing Semantic Pointer Architecture techniques.

Even with these limitations, this model presents an intriguing alternative to localist accounts of the Stroop effect. However, more work must be done to validate this as a cognitively plausible model.

References

Ashcraft, M. H. (1994). *Human memory and cognition.* New York, NY: Harper Collins.

Augustinova, M., Parris, B. A., & Ferrand, L. (2019). The loci of stroop interference and facilitation effects with manual and vocal responses. *Frontiers in psychology*, *10*.

Botvinick, M. M., Braver, T. S., Barch, D. M., Carter, C. S., & Cohen, J. D. (2001). Conflict monitoring and cognitive control. *Psychological review*, *108*(3), 624.

Bugg, J. M., Jacoby, L. L., & Toth, J. P. (2008). Multiple levels of control in the stroop task. *Memory & cognition*, *36*(8), 1484–1494.

Bustamante, L. A., Lieder, F., Musslick, S., Shenhay, A., & Cohen, J. (2020). Learning to overexert cognitive control in a stroop task.

Caron, E. E., Reynolds, M. G., Ralph, B. C. W., Carriere, J. S. A., Besner, D., & Smilek, D. (in press). Does posture influence the stroop effect? *Journal of Psychological Science*.

Choo, X., & Eliasmith, C. (2010, 08/2010). A spiking neuron model of serial-order recall. In R. Cattrambone & S. Ohlsson (Eds.), . Portland, OR: Cognitive Science Society.

Cohen, J. D., Dunbar, K., & McClelland, J. L. (1990). On the control of automatic processes: a parallel distributed processing account of the stroop effect. *Psychological review*, *97*(3), 332.

Cohen, J. D., & Huston, T. A. (1994). "progress in the use of interactive models for understanding attention and performance. *Attention and performance XV: Conscious and nonconscious information processing*, *15*, 453.

Corbetta, M., Miezin, F. M., Dobmeyer, S., Shulman, G. L., & Petersen, S. E. (1991). Selective and divided attention during visual discriminations of shape, color, and speed: functional anatomy by positron emission tomography. *Journal of neuroscience*, *11*(8), 2383–2402.

Dunbar, K., & MacLeod, C. M. (1984). A horse race of a different color: Stroop interference patterns with transformed words. *Journal of Experimental Psychology: Human Perception and Performance*, *10*(5), 622.

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. New York, NY: Oxford University Press.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems* [book]. Cambridge, MA: MIT Press.

Georgopoulos, A., Schwartz, A., & Kettner, R. (1986). Neuronal population coding of movement direction. *Science*, *233*(4771), 1416–1419. doi: 10.1126/science.3749885

Gosmann, J., Voelker, A. R., & Eliasmith, C. (2017). A spiking independent accumulator model for winner-take-all computation. In *Proceedings of the 39th annual conference of the cognitive science society.* London, UK: Cognitive Science Society.

Herd, S. A., Banich, M. T., & O'reilly, R. C. (2006). Neural mechanisms of cognitive control: An integrative model of stroop task performance and fmri data. *Journal of cognitive neuroscience*, *18*, 22–32.

Kajić, I., Gosmann, J., Stewart, T. C., Wennekers, T., & Eliasmith, C. (2017). A spiking neuron model of word associations for the remote associates test. *Frontiers in Psychology*, *8*, 99. doi: 10.3389/fpsyg.2017.00099

Kajić, I., Schröder, T., Stewart, T. C., & Thagard, P. (2019). The semantic pointer theory of emotion: Integrating physiology, appraisal, and construction. *Cognitive Systems Research*. doi: https://doi.org/10.1016/j.cogsys.2019.04.007

Levy, S. D., & Gayler, R. (2008). Vector symbolic architectures: A new building material for artificial general intelligence. In *Proceedings of the 2008 conference on artificial general intelligence 2008: Proceedings of the first agi conference* (p. 414–418). NLD: IOS Press.

Lovett, M. C. (2005). A strategy-based interpretation of stroop. *Cognitive Science*, *29*(3), 493-524. doi:

10.1207/s15516709cog0000_24

Lu, T., Voelker, A. R., Komer, B., & Eliasmith, C. (2019). Representing spatial relations with fractional binding. In *41st annual meeting of the cognitive science society.* Montreal, QC: Cognitive Science Society.

MacLeod, C. M. (1991). Half a century of research on the stroop effect: an integrative review. *Psychological bulletin*, *109*(2), 163.

Petersen, S. E., & Posner, M. I. (2012). The attention system of the human brain: 20 years after. *Annual review of neuroscience*, *35*, 73–89.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623-641.

Rayner, K., & Pollatsek, A. (1989). *A.(1989). the psychology of reading englewood cliffs.* NJ: Prentice-Hall.

Rumelhart, D. E., McClelland, J. L., & PDP Research Group, C. (Eds.). (1986). *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: Foundations*. Cambridge, MA, USA: MIT Press.

Stewart, T. C., Bekolay, T., & Eliasmith, C. (2011). Neural representations of compositional structures: Representing and manipulating vector spaces with spiking neurons. *Connection Science*, *23*(2), 145–153.

Stewart, T. C., & Eliasmith, C. (2012). Compositionality and biologically plausible models. In W. Hinzen, M. Werning, & E. Machery (Eds.), *Oxford handbook of compositionality.* Oxford University Press.

Stroop, J. R. (1935). Studies of interference in serial verbal reactions. *Journal of experimental psychology*, *18*(6), 643.

Using a Bidirectional Associative Memory and Feature Extraction to model Nonlinear Exploitation Problems

Kinsey Church (kchur026@uottawa.ca)
School of Psychology, 136 Jean-Jacques Lussier, Vanier Hall
Ottawa, ON, K1N 6N5, CAN

Matt Ross (mross094@uottawa.ca)
School of Psychology, 136 Jean-Jacques Lussier, Vanier Hall
Ottawa, ON, K1N 6N5, CAN

Sylvain Chartier (sylvain.chartier@uottawa.ca)
School of Psychology, 136 Jean-Jacques Lussier, Vanier Hall
Ottawa, ON, K1N 6N5, CAN

Abstract

Each day we are faced with a decision of maximizing our resources by using our current knowledge to learn new things. Should we go to the new restaurant that just opened around the corner or stick to an old, reliable favourite? This is known as the exploration-exploitation dilemma and it is at the heart of reinforcement learning. The present study looks at the exploitation half of this problem and aims to implement it in a biologically plausible recurrent associative memory model. In the framework of Artificial Neural Networks, exploitation is observed when the network can iterate through many learned responses and stabilize on the correct one to solve a given task. This is a process akin to being able to switch from a cyclic to a point attractor. More precisely, Bidirectional Associative Memory (BAM) is used to accomplish such tasks where the context dictates which attractor the network should converge to. For simple independent tasks, the BAM is sufficient. However, for overlapping tasks, the task becomes nonlinearly separable. Therefore, the BAM needs an extra unsupervised layer to extract unique features from the inputs. These features combined with input are then sent to BAM where it can learn the different attractors adequately. This network was able to stabilize on the correct responses of tasks that involved time series of varying lengths, overlap, and levels of correlation; the variability one would expect from the real world.

Keywords: exploitation/exploration; recurrent associative memory; one-to-many associations.

Introduction

In order to increase their chances of survival, many organisms have evolved various complex mechanisms to solve problems, make decisions, and learn about the environment around them. These mechanisms come with several different trade-offs, where increasing proficiency in one area may result in a lack of ability in another area. One of these problems in cognition is the exploration-exploitation dilemma (Hills, Todd, Lazer, Redish, & Couzin, 2015). This is when an organism must choose whether it wants to employ something it has already learned (exploitation) or explore its environment to learn new things (exploration). For example, when a young child is first learning how to open different doors. They may start by trying the few ways they already know to open a door (exploitation), such as pulling and pushing. If these behaviours fail, however, they will have to start looking for new solutions (exploring). Understanding exactly how animals and humans switch between these two processes and how each of them works is crucial to understanding how we adapt, learn, behave, and survive (Hills et al., 2015). This trade-off is one of many phenomena affected by Reinforcement Learning (RL).

Previous studies have proposed several methods to solving this problem from an "optimal solution" perspective. For example, Hwang, Chiou and Wu (2003) created an Artificial Neural Network (ANN; an evaluation predictor) that solves the exploration-exploitation problem that was able to both explore and exploit to find an optimal solution to their task (balancing a pendulum). They used the mean of a normal distribution to represent the exploitation function, which may be a bit too basic to capture the exploitation process. This study and many others focus too much on obtaining a desired state rather than the underlying process of how exploitation (and exploration) take place or attempt to find a biologically plausible solution (Cohen, McClure, & Yu, 2007; Tilahun, 2019).

Another promising study was conducted by Lew, Rey, and Zanutto (2013). It outlines a biologically plausible model of switching between exploration and exploitation depending on changes in the environment and rewards from the dopaminergic system. Their model is a computational framework of how exploration and exploitation are selected and relates this process back to the reward system in the human brain, focusing on the selection only. Many studies have a similar focus and are more about the overall model rather than how each process works (Gershman & Niv, 2015; Hallquist & Dombrovski, 2019).

An important consideration overseen by these studies is the context in which the decision takes place. Empirical studies have shown how the environment is an important factor in predicting how an animal will behave and how it will exploit its resources (Naruse et al., 2018). Some studies have shown how the decisions humans and animals make influence their ecosystem and vice versa, from small-scale interactions

among individual agents to large-scale adaptive systems, like population dynamics (Monk et al., 2018). The context in which an animal is found clearly plays a role on how it chooses to exploit and explore.

The present study looks at the exploitation half of the dilemma using environmental context as a cue. For instance, an animal faces the exploration-exploitation dilemma while foraging for food (Hills et al., 2015). Is it more efficient to explore the area for new sources of food or stick to a learned set of locations where food has been found in the past?

The most well-known example of exploitation has been in Q-Learning (Watkins & Dayan, 1992). In this implementation, the best outcome is simply the one that has the maximum possible reward; max(Q). This works best when the reinforcement is non-binary. However, in the situation of multiple possible outcomes, the agent should be able to test different possible behaviours until the correct one has been found. For example, if the animal chooses to exploit their already-known resources, they cycle through them until the problem is solved (they find food). Once it is solved, they will stop trying different behaviours and focus on the successful one.

In ANNs, the problem can be framed as how the network can generate a different output (behaviour) when the input (context) remains the same? In ANNs, if a given input is sent to the network, it will always give the same associated output. However, if that output is not satisfying, the network must give a different output even though it is still the same input that is presented. One way to solve this problem is to use a cycle attractor. This process can be modeled by the network generating various different outputs from the same input to represent the different learned behaviours. For example, input 1 generates output 1, which generates output 2, all the way until output *n*. Output *n*, the final output, is then associated with the output 1, creating a loop of learned patterns. This would be like a squirrel having a series of berry bushes where it knows food can be found and going from bush to bush until it finds some.

This can be easily implemented in a Bidirectional Associative Memory (BAM) as a multistep time series (Chartier & Boukadoum, 2006). However, a problem arises when the network must stabilize on a specific output *i*. Therefore, each of the patterns (output) must also be stored as point attractors in the network. This represents a one-to-many situation where for the same output *i* the network must generate output *i*+1 (next pattern; cyclic attractor) or output *i* (same pattern; point attractor). See Fig. 1 for a diagram of cyclic and point attractors.

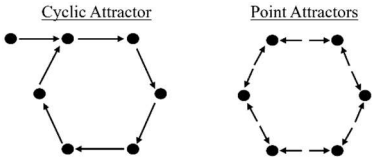


Figure 1: Illustration of cyclic and point attractors.

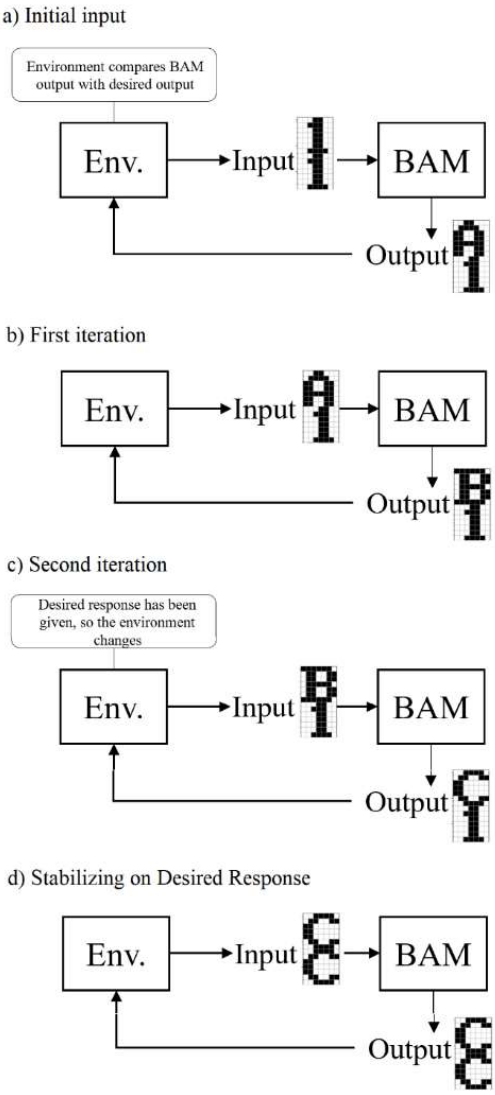


Figure 2: Flowchart illustrating the switching from a cyclic attractor to a fixed-point attractor using the environment.

The solution may reside in the inclusion of context that can dictate which state the network should be in and therefore give distinctive features. From the feedback given by the environment, the network should be able to switch between a cyclic (repeating series of outputs) and a point attractor (single output). However, instead of using context from the time series itself (Elman, 1990), the context has to be extracted from the reading of the environment; a process proposed in (Jordan, 1997) and successfully implemented in BAM (Rolon-Mérette, Rolon-Mérette, & Chartier, 2019). By having a unique context as the initial input of the time series, it can also be used to distinguish multiple cyclic attractors (different time series).

The BAM can be used to store various attractors including point, cyclic, and chaotic attractors (Chartier, Boukadoum, & Amiri, 2009), making it a well-suited candidate to perform exploitation. Fig. 2 shows an example of both cyclic

attractors (a to c) and fixed-point attractors (d) in the model. When the BAM is giving the next output, it is recalling associations in a cycle. In this case, it recalls 'A' from '1' and proceeds in order ('B', 'C', etc.). However, when it needs to stabilize, it switches to a fixed-point attractor. In this example, the environment gives 'C' associated with itself as a response, which is different from 'C' in the '1' context, causing the BAM to switch from a cyclic attractor to a fixed-point attractor. In order to focus exclusively on the exploitation phase, it is assumed that some level of exploration has already taken place. In other words, that series of behaviours have already been learned.

Putting Fig. 2 in the context of the foraging example, '1' could represent hunger, and the other patterns represent learned behaviours, such as searching for food in different locations; 'A', 'B', 'C', etc. When food is located, the environment lets the animal know to stop cycling through its learned behaviours (checking different locations) and make the decision to stay there.

The remainder of this paper is divided into two separate sets of simulations. Simulation I introduces the problem of modeling exploitation and switching from iteration to stabilizing on a given response using the environmental context (one-to-many association). Simulation II solves more complex and diverse situations involving nonlinearly separable associations. An overall discussion is provided at the end.

Simulation I- Independent Time Series

Methods

To model exploitation, time series of patterns were used to represent different possible output behaviours. Each series was preceded by a unique contextual input. The network task consisted of associating multiple time series of different lengths and various levels of overlap. Moreover, the level of correlation between patterns varies vastly to encapsulate the variability found in the real world.

Stimuli Patterns of behaviours were represented by a series of letters and the context by a number. Each pattern was placed on a 7x7 grid where a black pixel had a value of +1 and a white pixel a value of -1. Following the procedure in Rolon-Mérette et al. (2019) for each pattern of the time series, the associated context pattern was appended. These contexts allow the network to differentiate between different time series and the desired type of attractor (cyclic vs. point). The various time series are illustrated in Fig. 3, with grey boxes to highlight the repeated and overlapping patterns. Series 1 is a single time series of patterns associated with the context of number '1' and each of the patterns are also associated with themselves. It represents a basic, independent time series with no overlap and is the least complex series. Series 2 increases the difficulty by having two independent time series of various lengths. Series 3 includes three independent time series of various lengths as well. Series 4 and 5 represent situations closer to reality, where the difficulty is further increased. In both examples, a given pattern is associated not only to two settings (the cyclic and the point attractor) but to four (series 4) or three (series 5) settings. Series 4 shows a setting where patterns can belong to various solutions (overlapping), while series 5 shows a series that is a subtype of the other. In both cases, the problem is no longer linearly solvable.

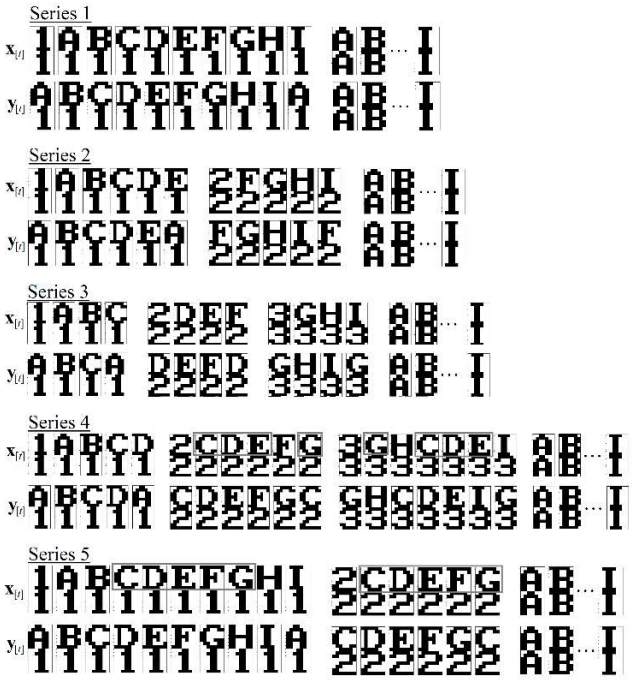


Figure 3: Stimuli for Simulations I and II.

Architecture The architecture for the modified BAM (Chartier et al., 2006) is made of two Hopfield-like neural networks interconnected in a head-to-tail fashion to create a bidirectional flow of information (see Fig. 4). The initial vector-states are represented by $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$, the weight matrices by **W** and **V**, the dimensionality by *m* and *n*, and *t* represents the time (current iteration number).

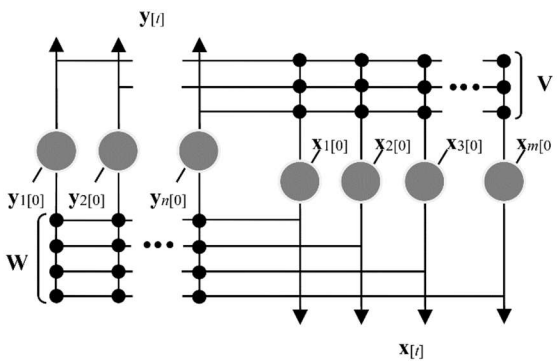


Figure 4: Architecture of the BAM.

Output Function The transmission is composed of the activation function that is then fed through the output function. The activation is obtained the usual way:

$$\mathbf{a}_{[t]} = \mathbf{W}\mathbf{x}_{[t]} \tag{1a}$$

$$\mathbf{b}_{[t]} = \mathbf{V}\mathbf{x}_{[t]} \tag{1b}$$

Once this linear activation is obtained it goes through the nonlinear output function defined as (2a) and (2b):

$$\forall i,\ldots,n,\ \mathbf{y}_{i[t+1]} = f(\mathbf{a}_{i[t]}) = \begin{cases} 1, & \text{if } \mathbf{a}_{i[t]} > 1 \\ -1, & \text{if } \mathbf{a}_{i[t]} < -1 \\ (\delta+1)\mathbf{a}_{i[t]} - \delta\mathbf{a}_{i[t]}^3, & \text{else} \end{cases} \tag{2a}$$

$$\forall i,\ldots,m,\ \mathbf{y}_{i[t+1]} = f(\mathbf{b}_{i[t]}) = \begin{cases} 1, & \text{if } \mathbf{b}_{i[t]} > 1 \\ -1, & \text{if } \mathbf{b}_{i[t]} < -1 \\ (\delta+1)\mathbf{b}_{i[t]} - \delta\mathbf{b}_{i[t]}^3, & \text{else} \end{cases} \tag{2b}$$

where n and m are the number of units in each layer of the network, i is the index unit, δ is the transmission parameter, and **a** and **b** are the activations. The output is a cubic function with saturating limits at ±1.

Learning Function The connection weights are modified following a Hebbian/anti-Hebbian rule (Bégin & Proulx, 1996; Storkey & Valabregue, 1999) and can be represented as:

$$\mathbf{W}_{[k+1]} = \mathbf{W}_{[k]} + \eta(\mathbf{y}_{[0]} - \mathbf{y}_{[t]})(\mathbf{x}_{[0]} + \mathbf{x}_{[t]})^{\mathrm{T}} \tag{3a}$$

$$\mathbf{V}_{[k+1]} = \mathbf{V}_{[k]} + \eta(\mathbf{y}_{[0]} - \mathbf{y}_{[t]})(\mathbf{x}_{[0]} + \mathbf{x}_{[t]})^{\mathrm{T}} \tag{3b}$$

where **W** and **V** represent the two sets of weight connections, $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$ the initial inputs, η the learning parameter, and k the current learning trial. The η is calculated using the following inequality function to guarantee that the learning will converge (Chartier et al., 2006):

$$\eta < \frac{1}{2(1-2\delta)\mathrm{Max}[m,n]}, \delta \neq \frac{1}{2} \tag{4}$$

Parameters To ensure all associations are stable, delta (δ) was set to 0.2, number of iterations (t) to 1 and learning rate (η) was set to 0.008 (equation 4). The minimum Mean Squared Error (MSE) was 10^{-8} and the maximum amount of learning trials was set to 1000 (in case of non-convergence). The weights **W** and **V** were initialized at 0.

Learning Procedure

1. Selection of a series (Fig.3).
2. Selection of an associated pair as the initial input for $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$.
3. Computation of the outputs $\mathbf{x}_{[1]}$ and $\mathbf{y}_{[1]}$ using equations 1 and 2.
4. Weights update (**W** and **V**) according to equation 3.
5. This was repeated with each of the subsequent patterns until the MSE for all associations reached 10^{-8} or the number of trials reached 1000.

Exploitation Procedure

1. Selection of a "desired response" for the network to stabilize on for a given time series ("initial context"). This is the response stored in the environment for comparison.
2. First pattern in the time series is used as the input for BAM ($\mathbf{x}_{[0]}$).
3. Environment compares output ($\mathbf{y}_{[t]}$) with the "desired response" for each iteration. If it does not match, the network uses this pattern as the new input ($\mathbf{x}_{[t]}=\mathbf{y}_{[t]}$) and generates the next one ($\mathbf{y}_{[t+1]}$). If it matches (BAM has found the correct pattern), the environment sends only this pattern back to the BAM. This can be seen as the environment changing when the solution to a task is found (for example, when you are trying to open a door and it opens vs. remaining closed), providing enough feedback to stabilize on this correct response.
4. The BAM recalls the pattern associated with itself and stabilizes on it instead of continuing to iterate through the series.

Results

The BAM was able to learn and recall the first three series. It was also able to stabilize its output on any desired pattern with perfect accuracy. For example, in Fig. 5 we see one trial in series 1, where 'F' was the correct response in time series '1'. The initial input was '1' and it was given to the BAM, which recalled 'A'. This output was compared to the desired response in the environment, and since it was not a match, the output became the new input and was sent back to the BAM. This continued until the correct response 'F' was given, where the environment changed because the task had been solved and sent 'F' both as input and context, to the BAM.

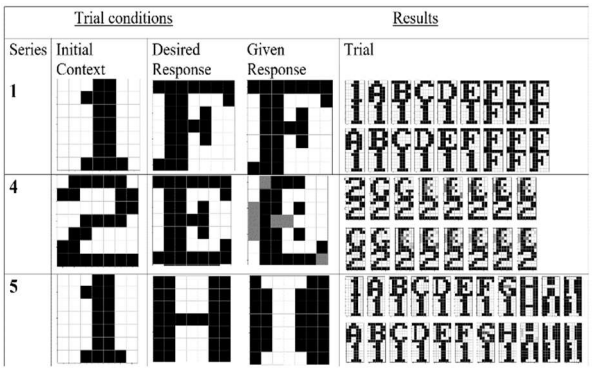


Figure 5: Results of example trials from Simulation I.

The network was then able to give the same output, therefore causing it to stabilize on the correct response.

However, when the BAM was faced with more complex tasks (overlaps or subtype), it could no longer learn the associations properly and stabilize on the desired response. For example, when encountering a nonlinear task as seen in series 4 and 5, the results are noisy, and the BAM often fails to learn all the time series properly. It also cannot stabilize on the correct response. This is due to the nature of the task that necessitates a nonlinear classification, a capacity lacking in a single layer BAM (Chartier & Boukadoum, 2006). Therefore, the next simulation solves this problem by adding an unsupervised layer to the BAM (Rolon-Mérette, Rolon-Mérette, & Chartier, 2018).

Simulation II- Overlapping Time Series

Methods

As indicated by the results from Simulation 1, a BAM alone using contexts is not sufficient to solve problems of higher complexity, meaning it cannot solve real-world problems. Previous works have shown that a Feature Extraction Bidirectional Associative Memory (FEBAM) can be used in combination with the BAM in order to solve nonlinearly separable tasks (Tremblay, Myers-Stewart, Morissette, & Chartier, 2013). Therefore, the architecture of the network has to be modified accordingly. The new flowchart is shown in Fig. 6.

Stimuli All series from Fig. 3 were used with the addition of one additional complex series (Fig. 7). This new series includes a mix of all possible combinations of multiple, overlapping and subtype time series; representing the kind of variability found in nature. It encompasses all possible combinations and arrangements of stimuli. Fig. 8 illustrates these combinations, using a Venn diagram for clarity.

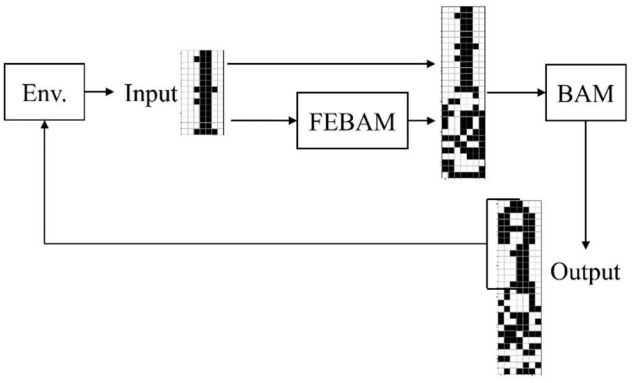


Figure 6: Flowchart for Simulation II, showing unique representation from FEBAM being appended to stimuli.

Figure 7: Added stimuli for Simulation II.

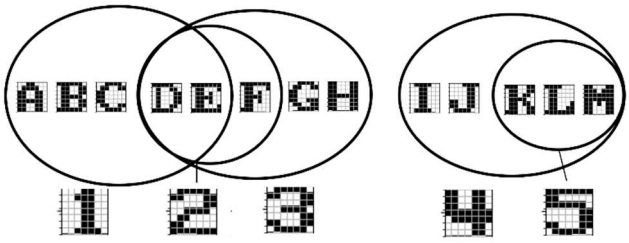


Figure 8: Venn diagram illustrating which time series are independent, subtype, or overlap in series 6.

Architecture The FEBAM is the unsupervised version of the BAM. The only difference between the two models is the absence of a set of external connections; the $\mathbf{y}_{[0]}$ inputs (see Fig. 4). This means there is only one explicit connection, $\mathbf{x}_{[0]}$, making the learning unsupervised.

Output Function and Learning Function The transmission is obtained the same way as the BAM. Regarding the learning, because the $\mathbf{y}_{[0]}$ connections are no longer available, they are obtained by iterating once through the network as illustrated in Fig. 9. The weights of the FEBAM are then updated exactly as the BAM (equations 3a, 3b).

Parameters For both BAM and FEBAM, the transmission parameter (δ) was set to 0.2 to ensure the fixed points would be stable. The learning parameter (η) was set to 0.004. The maximum amount of learning trials was set to 1000, and the minimum required MSE was 10^{-8}. The weights of the FEBAM were randomly initialized between -2 and 2 (Tremblay et al., 2013).

Learning Procedure

1. Selection of a series (Fig. 2 and 6).
2. Selection of an associated pair as the initial input for the FEBAM.
3. The output of the FEBAM is then appended to the initial input and sent to the BAM (Fig. 6).
4. The BAM outputs the next associated patterns.
5. Weights update for both the FEBAM and BAM.
6. This was repeated with each of the subsequent patterns until the MSE for all associations reached 10^{-8} or the number of trials reached 1000.

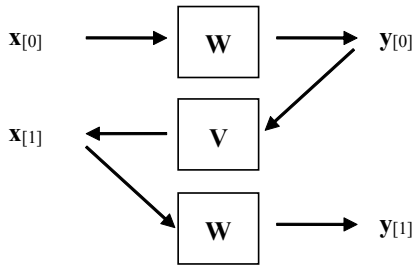


Figure 9: Output iterative process used for the FEBAM learning.

Exploitation Procedure

1. Selection of a "desired response" for the network to stabilize on for a given time series ("initial context"). This is the response stored in the environment for comparison.
2. First pattern in the time series is an input for FEBAM.
3. The output of the FEBAM is then appended to the initial pattern and sent to the BAM.
4. Environment compares output of the BAM (**y**[*t*]) with "desired response" for each iteration *t*. If it does not match, the network uses this pattern as the new input (**x**[*t*]=**y**[*t*]) and generates the next one (**y**[*t*+1]). If it matches (BAM has correctly solved the task), the environment sends only this pattern back to the FEBAM (then to the BAM).
5. The BAM recalls the pattern associated with itself and stabilizes instead of continuing to iterate through the series.

Results

As shown in Fig. 10, the FEBAM-BAM model was able to learn and perfectly recall both time series 4 and 5 with no problems, which the BAM alone was not able to do. It could also stabilize on any given response without issue. In the example shown for series 4 (Fig. 10), the desired response for that trial was 'E' in the '2' time series. The first input for that time series, '2', was given as the initial input and sent through the FEBAM, where the unique signature for that pattern was generated and appended to it. This new stimulus was then entered as input for the BAM, which gave 'C' as an output. That output was compared to the desired response in the environment and was sent back to the FEBAM until the desired response, 'E' was given. This desired response 'E' was then sent, associated with itself, as the new input for FEBAM, causing the BAM to stabilize on the correct answer. Most importantly, the model was able to successfully stabilize on any given response in series 6, which contained all possible combinations of interaction and correlation between stimuli: subtype, independent time series, and overlap. This model was successfully able to iterate through all series and stabilize on a given response.

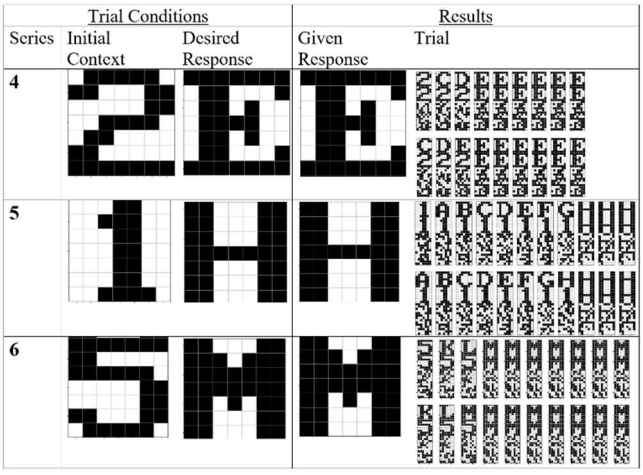


Figure 10: Results of example trials from Simulation II.

Discussion and Conclusion

To further our understanding of Reinforcement Learning and the exploration-exploitation trade-off, the exploitation phase was implemented using a BAM. Here it was assumed that a given set of patterns were already encoded for various contexts. The network's job was then to iterate through a given series and stabilize on the desired response using feedback from the environment. Various levels of complexity were tested ranging from a single time series all the way to the same variability found in a natural setting. When the complexity was too difficult for a single BAM (nonlinearly separable cases), a FEBAM was included in order to generate unique representations for each pattern. This combination of networks was sufficient to solve any type of situation.

Furthermore, by adding layers of FEBAM-BAM, the point attractors could become the context of a new time series and allow chains of time series; something akin to chunking (Gobet et al., 2001).

One limitation of this study is that it represents exploitation under very specific circumstances. For example, if one is trying to solve a problem with a series of learned responses, they will learn which of the responses are helpful and which are not. Therefore, the order of the patterns should reflect the probability of success, a phenomenon well-captured by standard Q-Learning. Changing the order of some items without retraining the whole dataset is a challenging avenue in a distributed associative memory and should be addressed in future studies. This could be implemented using an additional BAM to store "correct" responses in function of the desired new order with a novel correlated context generated by the network itself or by switching to more interesting attractors such as aperiodic ones (Tsuda, 2001).

The current network could also be added to a model of exploration to study the exploration-exploitation trade-off. Finally, temporal aspects should be taken into consideration, allowing for the timing of when actions should be accomplished, easing the transition from numerical simulation to real-time neurorobotic implementation.

In conclusion, by using a BAM, we were able to model the exploitation phase of the exploration-exploitation dilemma,

where the subject iterates through different learned responses and can stabilize on the correct response based on feedback from the environment. By adding a FEBAM to generate a unique representation for each learned stimulus, we were able to model this with all of the complexity of a real-world setting, including different lengths of stimuli, different levels of correlation, and nonlinear problems. Being able to model exploitation is a crucial part of understanding our own cognition and how we learn from the dynamic world around us.

References

Bégin, J., & Proulx, R. (1996). Categorization in unsupervised neural networks: The Eidos model. *IEEE Transactions on Neural Networks*, *7*(1), 147–154. https://doi.org/10.1109/72.478399

Chartier, S., & Boukadoum, M. (2006). A Sequential Dynamic Heteroassociative Memory for Multistep Pattern Recognition and One-to-Many Association. *IEEE Transactions on Neural Networks*, *17*(1), 59–68. https://doi.org/10.1109/TNN.2005.860855

Chartier, S., Boukadoum, M., & Amiri, M. (2009). BAM learning of nonlinearly separable tasks by using an asymmetrical output function and reinforcement learning. *IEEE Transactions on Neural Networks*, *20*(8), 1281–1292. https://doi.org/10.1109/TNN.2009.2023120

Cohen, J. D., McClure, S. M., & Yu, A. J. (2007). Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *362*(1481), 933–942.

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, *14*(2), 179–211.

Gershman, S. J., & Niv, Y. (2015). Novelty and Inductive Generalization in Human Reinforcement Learning. *Topics in Cognitive Science*, *7*(3), 391–415.

Gobet, F., Lane, P. C. R., Croker, S., Cheng, P. C. H., Jones, G., Oliver, I., & Pine, J. M. (2001, June 1). Chunking mechanisms in human learning. *Trends in Cognitive Sciences*, *5*, 236–243. https://doi.org/10.1016/S1364-6613(00)01662-4

Hallquist, M. N., & Dombrovski, A. Y. (2019). Selective maintenance of value information helps resolve the exploration/exploitation dilemma. *Cognition*, *183*, 226–243. https://doi.org/10.1016/j.cognition.2018.11.004

Hills, T. T., Todd, P. M., Lazer, D., Redish, A. D., & Couzin, I. D. (2015). Exploration versus exploitation in space, mind, and society. *Trends in Cognitive Sciences*, *19*(1), 46–54. https://doi.org/10.1016/J.TICS.2014.10.004

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. In *Advances in Psychology*, *121*, 471–495. https://doi.org/10.1016/S0166-4115(97)80111-2

Lew, S., Rey, H. G., & Zanutto, B. S. (2013). Neuronal mechanisms underlying exploration-exploitation strategies in operant learning. *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 1–6.

Monk, C. T., Barbier, M., Romanczuk, P., Watson, J. R., Alós, J., Nakayama, S., … Arlinghaus, R. (2018). How ecology shapes exploitation: a framework to predict the behavioural response of human and animal foragers along exploration-exploitation trade-offs. *Ecology Letters*, *21*(6), 779–793. https://doi.org/10.1111/ele.12949

Naruse, M., Yamamoto, E., Nakao, T., Akimoto, T., Saigo, H., Okamura, K., … Hori, H. (2018). Local reservoir model for choice-based learning. *PLoS ONE*, *13*(10).

Rolon-Merette, D., Rolon-Merette, T., & Chartier, S. (2018). Distinguishing Highly Correlated Patterns using a Context Based Aproach in Bidirectional Associative Memory. *Proceedings of the International Joint Conference on Neural Networks*, *July 2018*, *1-8.*

Rolon-Mérette, D., Rolon-Mérette, T., & Chartier, S. (2019). Learning and Recalling Arbitrary Lists of Overlapping Exemplars in a Recurrent Artificial Neural Network. In *Proceedings of the International Conference on Cognitive Modelling*, 186-191.

Storkey, A. J., & Valabregue, R. (1999). The basins of attraction of a new Hopfield learning rule. *Neural Networks*, *12*(6), 869–876. https://doi.org/10.1016/S0893-6080(99)00038-6

Tilahun, S. L. (2019). Balancing the Degree of Exploration and Exploitation of Swarm Intelligence Using Parallel Computing. *International Journal on Artificial Intelligence Tools*, *28*(3). https://doi.org/10.1142/S0218213019500143

Tremblay, C., Myers-Stewart, K., Morissette, L., & Chartier, S. (2013). Bidirectional Associative Memory and Learning of Nonlinearly Separable Tasks. In *Proceedings of the International Conference on Cognitive Modeling,* 420–425.

Tsuda, I. (2001). Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems. *Behavioral and Brain Sciences*, *24*(5), 793–810.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-Learning. *Machine Learning, 8*(3-4). 279-292.

Detecting Learning Phases to Improve Performance Prediction

Michael G Collins[1,2] (michael.collins.ctr@us.af.mil)
Caitlin Tenison[3] (ctenison@ets.org)
Kevin A Gluck[1] (kevin.gluck@us.af.mil)
John Anderson[4] (ja0s@andrew.cmu.edu)

[1]Air Force Research Laboratory, [2]ORISE at AFRL, [3]ETS, [4]Carnegie Mellon University

Abstract

Models of learning and retention make predictions of human performance based on the interaction of cognitive mechanisms with temporal features such as the number of repetitions, time since last presentation, and item spacing. These features have been shown to consistently influence performance across a variety of domains. Typically omitted from these accounts are the changes in cognitive process and key mechanisms used by people while acquiring a skill. Here we integrate a model of skill acquisition (Tenison & Anderson, 2016) with the Predictive Performance Equation (PPE; Walsh, Gluck, Gunzelmann, Jastrzembski, & Krusmark, 2019) using Bayesian change detection (Lee, 2019). Our results show this allows for both better representation of an individual's performance during training and improved out-of-sample prediction.

Keywords: Mathematical model, Bayesian model, Hidden Markov model, Skill acquisition, Learning, Strategy change, Spacing effect, Change detection, Prediction

Since the research of Ebbinghaus over a century ago, psychologists have studied human learning and forgetting. This research has resulted in three core empirical phenomena. First, the law of practice (Newell & Rosenbloom, 1982). With sufficiently frequent practice on a task, performance improves over time. Second, the law of decay. As the time between instances of exposure increases, individual memory starts to decay and performance gets worse. Third, the spacing effect. When exposures between learning opportunities are distributed over time (spaced practice) individuals are slower to acquire the information, but retain the information better than if given the same number of exposures within a short time (massed practice; Carpenter, Cepeda, Rohrer, Kang, & Pashler, 2012). To explain and predict these factors of human learning, formal models of learning and retention have been developed (see Walsh, Gluck, Gunzelmann, Jastrzembski, & Krusmark, 2018, for a detailed model review and comparison). In the current paper, we focus on the Predictive Performance Equation (PPE), a model of learning and retention (Walsh et al., 2018).

Predictive Performance Equation (PPE)

PPE is a mathematical model of learning and retention that makes performance predictions at the individual level based on (1) prior performance and (2) the learning schedule of an individual. PPE is composed of five equations representing three psychological factors of learning (power law of learning, power law of decay, and spacing). The first factor is the power law of learning (Eq.1, first term), which is a function of *N*, the number of exposures to a task, *a* which represents one's prior task knowledge, and learning rate, *c*, which is held constant.

$$M = (N + a)^c * T^{-d} \ (Eq. 1)$$

The second factor is the power law of decay (Eq.1, second term). Temporal decay is represented using *T* (Eq. 2), which weights time by (Eq. 3), and exponent decay parameter, *d*.

$$T = \sum_{i=1}^{n-1} w_i \cdot t_i \ (Eq. 2)$$

$$w_i = t_i^{-x} \sum_{j=1}^{n-1} \frac{1}{t_j^{-.75}} (Eq. 3)$$

The third factor is spacing effect or temporal distribution of practice over time, which is represented within the decay parameter (Eq. 4).

$$d = b + m \cdot average\left(\frac{1}{log(lag_i)}\right) \ (Eq.4)$$

Spacing is accounted for using two free parameters *b* and *m* and cumulative average lag time. As practice events occur together (i.e., massed), the decay increases. As practice becomes distributed (i.e., spaced), the decay decreases. Finally, activation (M) is placed within a logistic function and adjusted according to a *threshold* parameter, τ.

$$Performance = \frac{1}{1 + exp\left(\frac{\tau - M}{s}\right)} \ (Eq. 5)$$

Model Limitations

A limitation of PPE is that it assumes an individual's performance is expected to improve or decrease according to a single continuous function over time. This is based on empirical findings of aggregate performance curves, which often reveal smooth performance curves following power laws. Research examining learning in individuals find this an artifact of averaging the performance of multiple individuals (Heathcote, Brown, & Mewhort, 2000; Gray & Lindstedt, 2017). An individual's performance can appear to have "sporadic" performance variation. These "sporadic" performance variations sometimes appear random but have been shown to reflect changes in an individual's strategy (Gray & Lindstedt, 2017; Tenison & Anderson, 2016)

These findings are consistent with research suggesting skill acquisition occurs in phases, where an individual uses

different strategies and/or representations to complete a task (Fitts & Posner, 1967; Tenison, Fincham, & Anderson, 2016). One such theory of skill acquisition, based on the ACT-R architecture, proposes that individuals go through three phases while acquiring a skill (Anderson, 1982). In the first phase, the *Computational phase*, the individual solves a problem by applying general problem-solving rules to achieve the solution. In the second, *Associative phase*, the problem is solved through the direct retrieval of various portions of the problem. In the third phase of learning, the *Autonomous phase*, the individual has created a stimulus-response rule for a given problem. Learning during these phases is driven by knowledge compilation and declarative strengthening. It can be modeled by three separate power law functions and their associated parameters (i.e, intercept, slope, asymptote). Prior work has used hidden Markov models (HMMs) to estimate these parameters and identify where phase shifts occur (Tenison & Anderson, 2016).

Currently, PPE does not represent or make predictions based on these learning phases. Prior research with PPE focused on accounting for average performance on memory retrieval during word association tasks and overall performance metrics of complex tasks (Gluck, Collins, Krusmark, Sense, Maaß, & van Rijn, 2019; Jastrzembski, Gluck, & Gunzelmann, 2006). In each of these cases, the assumption was a continuous performance curve moderated by features of the learning schedule (i.e., number of attempts, time between trials, spacing). Assuming a continuous learning curve is reasonable in these cases because average performance will follow standard features of learning. However, the argument for this assumption of a continuous performance curve weakens when accounting for performance at lower levels of aggregation. At the individual person learning individual skills level of analysis, there are often discontinuities in performance. PPE interprets these "sporadic" changes as noise, leading to (1) decreased confidence in an individual's performance, (2) a less accurate representation of the learning profile, and/or (3) unrealistic out-of-sample performance predictions. To mitigate these limitations a substantive mechanistic means of interpreting discontinuities in individual learning profiles is required.

In this paper, we propose and evaluate a theoretical and methodological integration to achieve just that. We use a Bayesian change detection procedure (Lee, 2019; Lee, Gluck, & Walsh, 2019) to identify when identifiable performance discontinuities occur. We then use information about these change points to infer changes in phase during learning and make predictions about subsequent performance. We refer to this novel implementation as TAPPED, which is an integration of Tenison and Anderson's (2016) skill acquisition model, Walsh et al.'s (2018) PPE, and Lee's (2018) change detection procedure. To foreshadow, the TAPPED model identifies similar learning phases similar to Tenison and Anderson's (2016) HMM-based models of skill acquisition and is found to have superior predictive accuracy compared to PPE.

Method

Participants

We used Amazon Mechanical Turk to recruit 101 participants (Gender: Female = 49, Age: M = 31.4, SD = 6). All participants were paid $10 for participation in both experimental sessions and $.02 for each correct problem.

Task Stimuli

During the experiment participants completed a set of novel mathematics problems (Pyramid problems). Each problem is composed of two numbers separated by a "$" symbol (e.g., 2$4). The first number is referred to as a base. The second number is referred to as the height. The base of the problem represents the first term in the additive sequence. The height of the problem represents the number of sequential numbers that must be added to the base. For example, if a participant was given the problem 3$3, then they would have to add together the number 3 + 4 + 5 + 6 = 18. In this experiment participants were given problems with bases ranging from 3 – 6 and heights from 4 – 11.

Procedure

The experiment consisted of two experimental sessions, with a 66 hour lag in between them. During the first day, problems were displayed on the screen and participants were instructed to type in their answer. All participants received feedback on whether they were correct or incorrect. Participants went through 10 practice blocks, with each block including 40 items each. Each block consisted of 40 items, with each item in one of four spacing conditions. Items in Spacing group 4 were presented 25 times with 3 problems in-between. Spacing group 8 were presented 25 times with 7 intervening questions. Spacing group 16 were presented 25 times with 15 intervening problems. Spacing group 32, were presented 12 times with 31 intervening presentations. The second experimental session was given 66 hours after the first session and were tested on the items they practiced on Day 1.

Hidden Markov Model

We fit an adaptation of the Tenison and Anderson (2016) power-law skill acquisition model to the response latencies for the items solved during the 10 practice blocks completed on Day 1. We refer to this model as the *Phase HMM*. Only an overview of the HMM model is provided here, a detailed description can be found in Tenison and Anderson (2016). The HMM consists of three states, each representing the three phases of skill acquisition. Within each phase, we have a state representing each practice opportunity the participant may have had within that phase. After each stimulus, participants either transition to the first state of the next phase or the next state within the current phase. The HMM predicts that participants' reaction times follow phase-specific power functions, where the opportunity count for each power function is determined by the current state within that phase. This HMM structure enables joint estimation of a participant's state as well as their response latency.

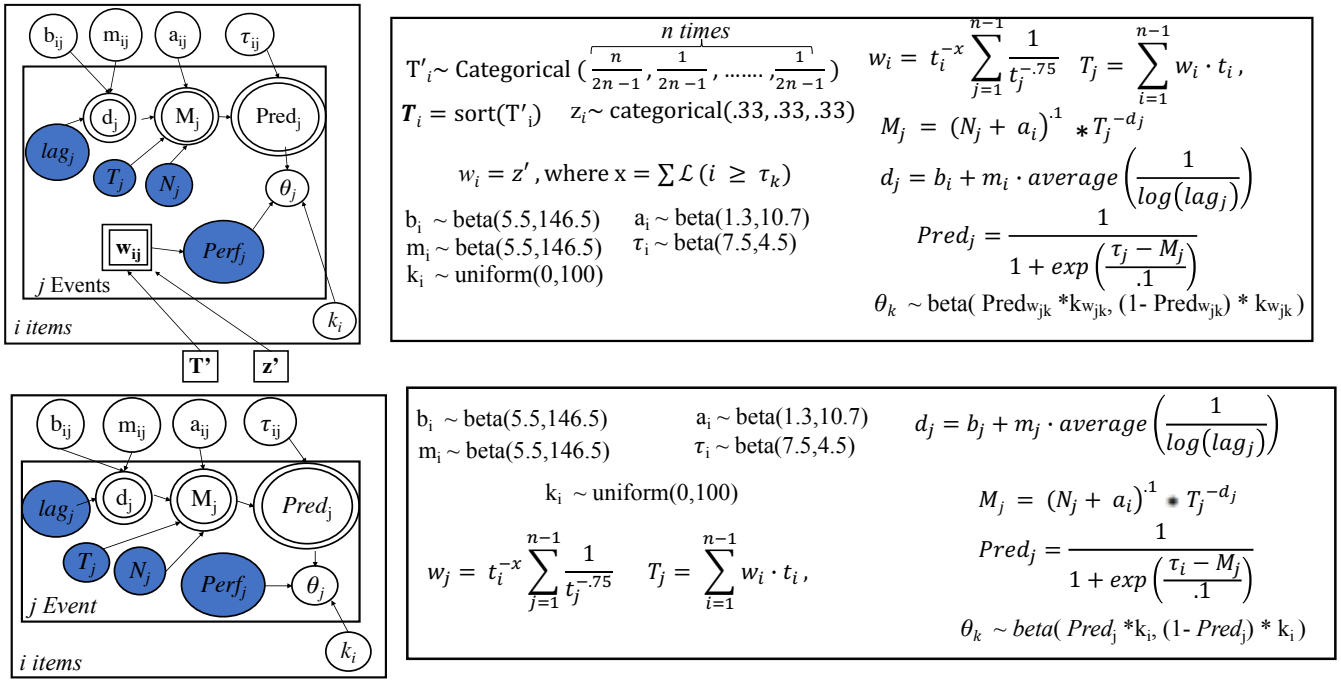


Figure 1. The Bayesian model diagrams for TAPPED (top) and PPE (bottom).

In fitting the Tenison and Anderson model (2016) to this current experiment we made two changes. First, we adjusted the model to account for general learning which occurs over the course of the task and is therefore shared across spacing groups (Eq 6). To do this we extended the skill specific model to capture general learning using the general learning equation derived from ACT* (Pirolli and Anderson, 1985). Second, we expanded our model fitting procedure to identify which parameters should be shared between items of different spacing groups. We fit eight models to the data, exploring whether sharing learning rate (α), transition probability (π_{12} π_{23}), and scale (β_{phase}) across spacing groups improved the fit of the model.

$$\mu_{ret} = \beta_{phase} n_{specific}{}^{-a_{specific}} \times n_{general}{}^{-a_{general}} \quad (Eq.6)$$

Table 1. Parameters for best fitting HMM

Spacing group	$\beta_{computation}$	$\beta_{associative}$	$\beta_{autonomous}$
4	9.97	3.6	1.8
8	11.5	4.6	2.0
16	12.0	5.1	2.1
32	12.0	4.6	1.9
Parameters shared across spacing group			
$a_{gen.}$	$a_{spec.}$	π_{12}	π_{23}
-.05	-.07	.12	.09

All models were compared using *BIC*. We found that the model that estimated unique scale parameters for each spacing group shared transition parameters and a shared learning rates fit best (Table 1; *BIC* = 119,517.2). This model fit the data better than more complex models with all unique parameters (*BIC*= 119,530.7) and simpler models with all shared parameters (*BIC* = 119,672.5). We list the parameters of our final model in Table 1. For each practice opportunity of each item, the model generates a likelihood of being in each state of the HMM. This can be translated into discrete Phase labels. We use these labels in our comparison with the TAPPED model.

Bayesian Models

The Bayesian implementations of TAPPED and PPE are each represented as a graphical model (Koller, Friedman, Getoor, & Taskar, 2007) (Figure 1). A graphical model format allows each variable, variable type, and the dependencies across variables to be observed. All observable variables (e.g., participant's response on a given trial – RT) are represented in shaded circles. PPE's estimated free parameters (*b*, *m*, *a*, τ) are represented as unshaded circles. Stochastic variables are represented with a single open circle, while deterministic variables are represented with two circles. The multiple panes represent redundancies for the different participants (i), and events (j).

The PPE (bottom graph) and TAPPED (top graph) (Figure 1) models are similar in their overall structure. Both models are run over a participant's (*i*) performance on a particular pyramid problem (item – *j*). For each participant the model estimates values for each of PPE's free parameter values (*b*, *m*, *a*, τ). The difference is in the number of different free parameters each model uses to account for participant performance during the 1st day. The PPE model (Figure 1 –

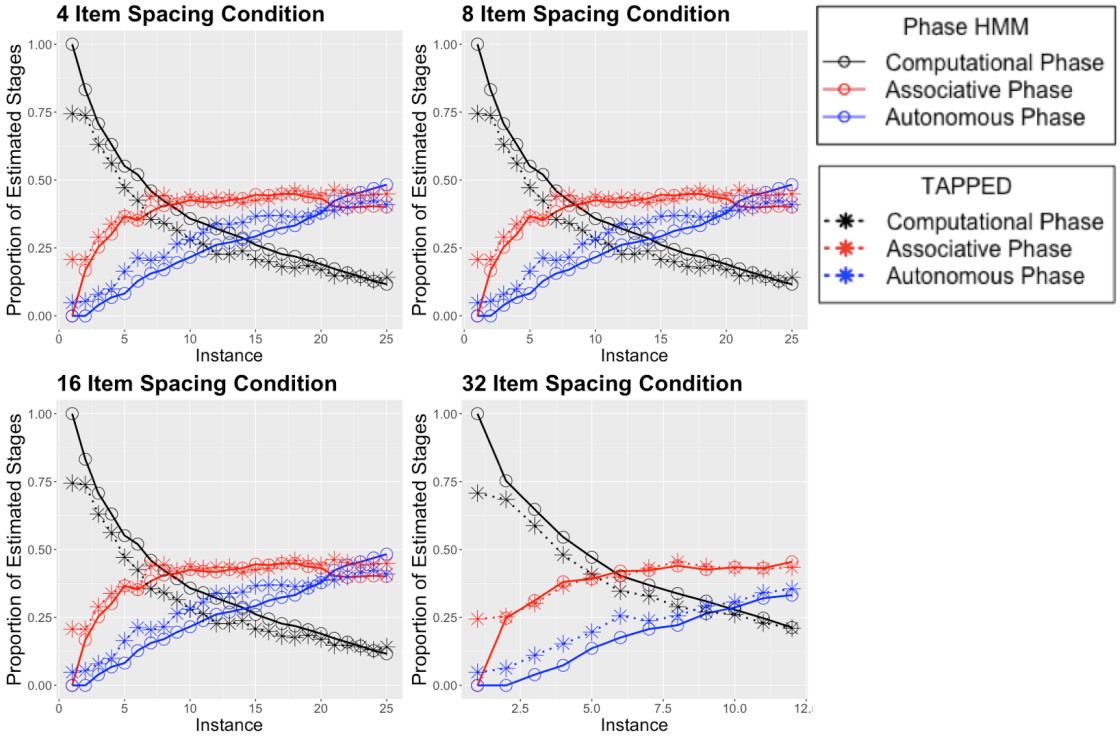


Figure 2. The round by round proportion of each phase of learning (Computational — black, Associative — red, and Procedural — blue) estimated with the Phase HMM (Open circles) and the TAPPED model (stars).

bottom figure) estimates only one value of each of PPE's free parameters for the participant's performance on a given item during the 1st day. However, the TAPPED model (Figure 2) can estimate up to 5 different values for each of PPE's free parameters, depending on the number of change points estimated by the model (*T*). Change points are estimated using a spike-and-slab prior (*T*) (Lee, 2019). After parameters are estimated for a portion of the learning curve (w_{ij}) from the prior distribution, they are combined with the participants' observed time variables (lag_j, *T*, N_j), and PPEs fixed equations

to create a performance estimate ($Pred_j$) for a given trial. The performance estimate ($Pred_j$) is then combined with a precision parameter (k_i) in a beta distribution to develop a prior distribution for the likelihood function. Because PPE is developed to predict accuracy, reaction times (*Perf*) were transformed to a proportion by dividing their reaction time by a maximum reaction time of 7 minutes. The opposite transformation was used to get PPE's predictions represent reaction time for all of the preceding results by multiplying PPE's predictions by 7 minutes.

Additionally, during the student's 1st day of performance, the TAPPED model also inferred the phase (i.e., computational, associative, procedural) of learning that each individual was in before and after each unique change point (*z*). Each of the three learning phases (declarative, associative, and procedural), was identified based on response times estimated from the Tenison, Fincham, and Anderson (2016) empirical data distribution. The estimations of the individual's phase were then used to make predictions of the participant's subsequent performance. Predictions the participants performance of the 2nd experimental session were generated by using parameters estimated for the highest learning phase obtained during Day 1 performance.

Results

First we will examine the similarity between the inferences of the participants' phases of learning estimated by Tenison and Anderson (2016) (Phase HMM) and our TAPPED model. A comparison between the two models' results allows us to evaluate the extent to which they reach converging inferences about learning phase. Second we will use the estimation of phases from the participants' performance during the 1st day to make theory-driven predictions of the 3rd day and compare these to the PPE.

Behavioral Effects

For this analysis, we only considered spacing groups with the same number of total practice opportunities (i.e. SG-4, SG-8 and SG-16). Rather than rely on accuracy, which is high across all spacing groups, we used response latency to verify the impact of spacing on performance. A repeated measures analysis of variance (ANOVA) run on mean latency data (log transformed) revealed a significant main effect of spacing group ($F(2,174)=97.6$, $p < .001$) and practice opportunity ($F(1,87)=684.7$, $p < .001$). The interaction between spacing group and practice was not significant ($F(2,174)=1.7$, $p = .2$). During the initial practice period, items that experience

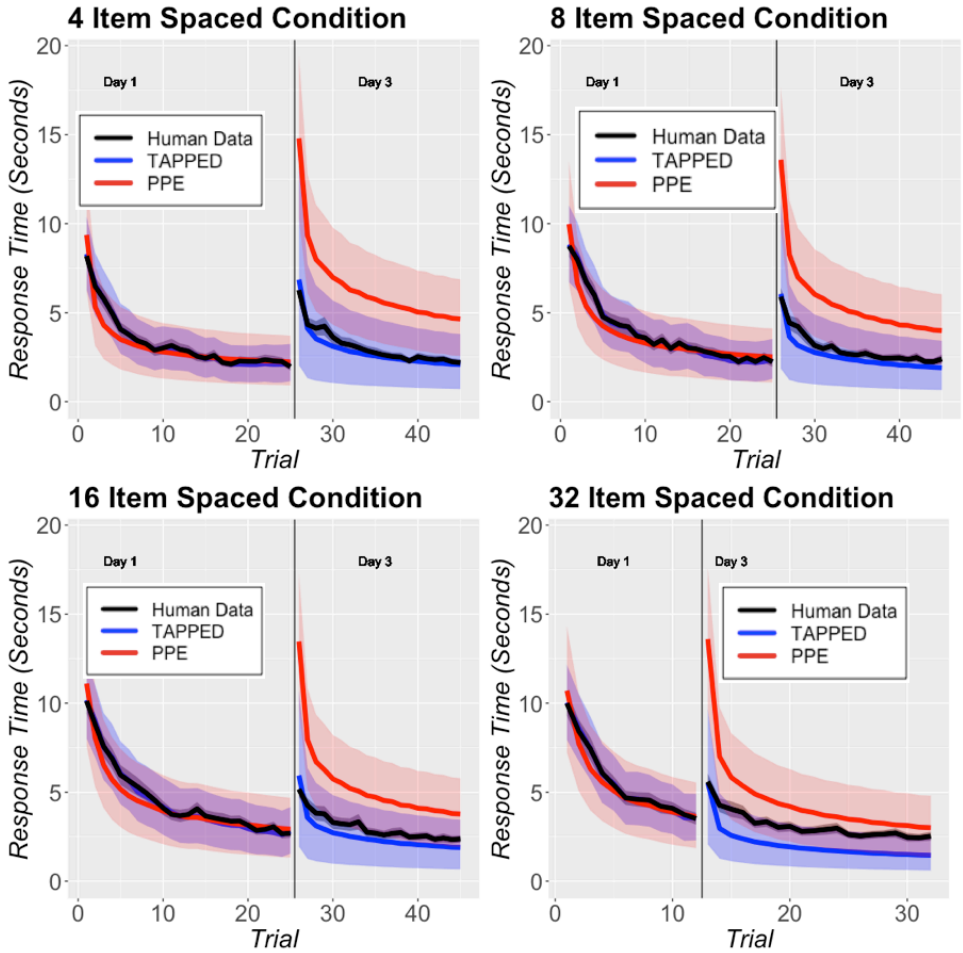


Figure 3. The mean performance of participants +/- 95% CI (black line and shaded ribbon) and the TAPPED +/- 95% HDI and PPE +/- 95% HDI calibration and prediction for the 4 spacing conditions.

An greater delay between practice opportunities take longer to solve than those with less delay between practice opportunities. We ran a repeated measures ANOVA to explore the impact of spacing on the response latency three days after the initial learning period. For the first opportunity on Day 3, we see a significant main effect of training ($F(2,174)=8.8, p<.001$) on problem solving latency (log transformed). The benefit of spaced practice is present in the faster response latency for spaced items, SG-4(M = 6.2s, SD = .3), SG-8 (M = 5.9s, SD = .3), SG-16 (M = 5.1s, SD = .3). These analyses confirm the presence of the spacing effect within our experiment.

Phase Comparison

To compare phases of learning inferred by the Phase HMM and the TAPPED model, the per round proportions of each phase (i.e., computational, associative, procedural) during the course of the first day in each of the 4 spacing conditions were compared (Figure 2). Across the four spacing conditions, a high degree of similarity is seen in the phases of learning estimated by the Phase HMM model and the TAPPED model (r = .87, RMSD = .10).

The greatest divergence between the two models is seen during the initial performance events (Rounds 1-3). Phase HMM model assumes that participants must sequentially go through all three phases of learning starting with the computational phase. This is why that model interprets 100% of participants as starting in the Computational phase. TAPPED does not share these assumptions, allowing for any phase of learning to be estimated at any point in time during the experiment for a given change point. Despite these differences, the TAPPED model converges to inferences similar to those of Phase HMM's model. This is evidence that the Bayesian change detection method of Lee (2019) is functionally approximating the output of the HMM used by the Phase HMM.

Day 1 – Calibration

To compare how well each model calibrated to each of the participant's performance during the 1st day, the PPE and TAPPED model are compared across the four spacing conditions (Figure 3) evaluation of each model's fit reveals two findings. First, the average performance estimate of both the PPE and TAPPED model fit participant average

performance quite well during the 1st day across each of the 4 spacing conditions.

Table 2. The correlation (*r*) and root mean squared deviation (RMSD) and the percent the participant's performance that falls within the predicted 95% HDI (% Pred) of the fits of TAPPED and PPE during the first experimental session.

	TAPPED			PPE		
Spacing	*r*	*RMSD*	*% Pred*	*r*	*RMSD*	*% Pred*
4	.93	1.32	93%	.76	2.14	84%
8	.94	1.46	92%	.77	2.69	82%
16	.93	1.68	90%	.77	2.87	77%
32	.98	1.05	96%	.81	2.99	77%

Although differences are observed in model fits of participants' individual learning curves, across the four spacing conditions, TAPPED has a higher correlation and lower RMSD with the individual participants compared to PPE (Table 2). TAPPED calibrates much more closely to individual performance during the 1st day. These better performance fits result from the fact that TAPPED selectively calibrates to separate portions of the participant's learning profile. While in contrast, PPE attempts to account for all the participant's first day performance with a single performance curve.

Day 3 – Predictions

Based on each model's calibration to the performance of participants during the 1st day, performance predictions were generated for each individual participant on the 3rd day (Figure 3). Over this period of time, the uncertainty each model has in the individual's performance increased. This uncertainty is reflected in the increase in the size of each model's 95% HDI. Though the uncertainty in predictions increases, there are differences between the two models' predictions. Over each of the 4 spacing conditions, PPE predicts slower initial and ongoing performance on the 3rd day – much slower than the human data. In contrast, TAPPED predicts faster initial and ongoing performance, much closer to the actual human experiment results.

Table 3. The correlation (*r*) and root mean squared deviation (RMSD) and the percent the participant's performance that falls within the predicted 95% HDI (% Pred) of the predictions of TAPPED and PPE during the second experimental session.

	TAPPED			PPE		
Spacing	*r*	*RMSD*	*% Pred*	*r*	*RMSD*	*% Pred*
4	.30	3.70	75%	.39	6.43	51%
8	.36	3.25	73%	.43	5.23	54%
16	.40	3.15	74%	.50	4.50	54%
32	.36	3.13	61%	.49	3.88	64%

The out-of-sample correlation and RMSD increase in both models across each of the 4 spacing conditions, relative to Day 1. PPE (Table 3) has a higher *r* but higher RMSD compared to TAPPED (Table 3). The overall decrease in the correlation between the models' predictions of the 3rd day is expected. However, in addition to how well the model captured learning during the 3rd day, we are interested in the accuracy of each model's predictions. To investigate this we calculated the percentage of response times that fell within a model's predicted 95% HDI. With the exception of the SG32 spacing condition, TAPPED had a greater predictive accuracy compared to PPE (Table 3). Here it is seen that TAPPED, despite having a lower correlation compared to PPE, was able to better predict the participants' actual performance data. This result suggests TAPPED's additional complexity is warranted, given its ability to predict out-of-sample performance.

Discussion

The results presented in this paper revealed several interesting findings. First, a comparison of the inferences of learning phase by TAPPED to Phase HMM during the 1st day of performance found a high degree of similarity. Differences between the two model's inferences about learning phases stemmed from the assumptions about the sequences of phase transitions over time. This particular Phase HMM assumes a strict sequential transition between learning phases and does not allow for regression back to previous learning phases. In contrast, TAPPED holds no such assumptions. These differences between the two models' assumptions lead to a particular disagreement in inferences of the participant's initial phases of learning during the 1st several trials. This added flexibility of TAPPED provides a more realistic model of skill acquisition in which forgetting can occur between problems. While this matters less in a highly focused training paradigm where forgetting is less likely, in more spaced and varied training paradigms this is a strength over the Tenison and Anderson (2016) model. Despite the differences in the assumptions of learning phase transitions between the two models, the high degree of similarity of the classification of learning phase over the 1st day suggest that both models are capturing similar aspects within the data.

We also compared TAPPED to PPE, contrasting how each model accounts for participant performance during the 1st day and predicting participant performance on the 3rd day. TAPPED was able to better fit participants' performance during the 1st day compared to PPE. This is due to the fact that the TAPPED has a greater number of parameters and was able to selectively fit to the performance curve of the individual. Despite TAPPED's additional complexity, this model better predicts performance on the 3rd day. This increase in accuracy comes from the fact that the TAPPED model demarcated and classified changes in a participant's performance and used information from the participant's most recent stage of learning to make a prediction.

In contrast, PPE developed predictions based on all of a participant's data from the first day, leading to predictions of much slower performance on Day 3. The only exception to these regularities were seen in the longest spacing conditions, where the performance of both models is nearly equal. The

improved performance in PPE could have been due to the nature of the spacing manipulation. The highly spaced nature of the 32-item spacing condition could have decreased abrupt changes due to changes in phase in an individual's performance allowing for the PPE to better capture and predict performance during the 2nd experimental session. However, further exploration is needed and future research should address the similarities and differences between TAPPED and PPE when fitting and predicting items on a longer spaced schedule. Furthermore, future research should investigate using TAPPED in a more complex learning tasks where individuals might go through successive iterations of the three learning phases addressed in this paper or vary in their proficiency in which they learn a particular skill. Understanding these subtle fluctuations or differences in performance is important for being able to predict performance at the individual item level.

In summary, models of learning and retention often account for performance at a given time based on an individual's prior performance and temporal features of study and practice history. Often these models do not represent the cognitive mechanisms or changes in cognitive mechanisms individuals use when acquiring a particular skill and how these particular mechanisms might interact with the presentation history of the learned material. Our results suggest that detecting and modeling learning phases can improve predictive validity.

Acknowledgements

This research was supported by National Science Foundation grant DRL-1420008 and by the U. S. Air Force Research Laboratory's 711th Human Performance Wing, through the Personalized Learning and Readiness Sciences Core Research Area. MGC's participation was enabled through an appointment to the Oak Ridge Institute for Science and Education (ORISE) Student Research Participation Program. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government..

References

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, 89, 369–406.

Carpenter, S. K., Cepeda, N. J., Rohrer, D., Kang, S. H., & Pashler, H. (2012). Using spacing to enhance diverse forms of learning: Review of recent research and implications for instruction. *Educational Psychology Review*, *24*(3), 369-378.

Fitts, P. M., & Posner, M. I. (1967). *Human performance*. Belmont, CA: Brooks/Cole

Gray, W. D., & Lindstedt, J. K. (2017). Plateaus, dips, and leaps: Where to look for inventions and discoveries during skilled performance. *Cognitive science*, *41*(7), 1838-1870.

Gluck, K. A., Collins, M. G., Krusmark, M. A., Sense, F., Maaß, S., & van Rijn, H. (2019). Predicting performance in cardiopulmonary resuscitation. In Stewart, T. C. (Ed.). Proceedings of the 17th International Conference on Cognitive Modeling (pp. 53-58). Waterloo, Canada: University of Waterloo.

Heathcote, A., Brown, S., & Mewhort, D. J. K. (2000). The power law repealed: The case for an exponential law of practice. *Psychonomic bulletin & review*, *7*(2), 185-207.

Jastrzembski, T. S., Gluck, K. A., & Gunzelmann, G. (2006). Knowledge tracing and prediction of future trainee performance. In the *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference* (pp. 1498-1508). Orlando, FL: National Training Systems Association.

Koller, D., Friedman, N., Getoor, L., & Taskar, B. (2007). Graphical models in a nutshell. *Introduction to statistical relational learning*, *43*.

Lee, M. D. (2019). A simple and flexible Bayesian method for inferring step changes in cognition. *Behavior research methods*, *51*(2), 948-960.

Lee, M. D., Gluck, K. A., & Walsh, M. M. (2019). Understanding the complexity of simple decisions: Modeling multiple behaviors and switching strategies. *Decision, 6*(4), 335–368

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R Anderson (Ed.), *Cognitive skills and their acquisition* (pp. I-55). Hillsdale, NJ: Erlbaum.

Pirolli, P. L., & Anderson, J. R. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology/Revue canadienne de psychologie*, *39*(2), 240.

Tenison, C., & Anderson, J. R. (2016). Modeling the distinct phases of skill acquisition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 42(5), 749–767.

Tenison, C., Fincham, J. M., & Anderson, J. R. (2016). Phases of learning: How skill acquisition impacts cognitive processing. *Cognitive psychology*, *87*, 1-28.

Walsh, M. M., Gluck, K. A., Gunzelmann, G., Jastrzembski, T., & Krusmark, M. (2018). Evaluating the theoretic adequacy and applied potential of computational models of the spacing effect. *Cognitive science*, *42*, 644-691.

Cognitive Salience of Features in Cyber-attacker Decision Making

Edward A. Cranford (cranford@cmu.edu)
Department of Psychology, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213 USA

Sterling Somers (sterling@sterlingsomers.com)
Department of Psychology, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213 USA

Konstantinos Mitsopoulos (cmitsopoulos@cmu.edu)
Department of Psychology, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213 USA

Christian Lebiere (cl@cmu.edu)
Department of Psychology, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213 USA

Abstract

While much is known about how humans make decisions based on the recency, frequency, and similarity of past experiences, much less is known about how humans weigh the contextual features and the impact it has on decisions. The present study uses a novel method of introspecting into a cognitive model of human decision making in an abstract cyber security game to gain insight about the cognitive salience of the features. The results show that cognitive salience can provide valuable evidence about how and why individuals make their decisions. The implications of these results are discussed with regard to theory and application.

Keywords: cognitive salience; cyber deception; cognitive models; instance-based learning, ACT-R

Introduction

Feature representation plays an important role in human decisions, and while much is known about how experiences shape decisions through instance-based learning (e.g., frequency, recency, and similarity effects; Gonzalez, 2013), much more needs to be understood about what features are represented in human decisions and how those features weigh in the decision. Instance-based learning (IBL) models have accurately modeled human behavior across a number of tasks including supply chain management (Gonzalez & Lebiere, 2005), social dilemmas (Lebiere, Wallach, & West, 2000), two-person games (Sanner et al., 2000, West & Lebiere, 2001), repeated binary-choice decisions (Gonzalez & Dutt, 2011), and multi-stage Stackelberg Security games (Cranford et al., 2020a). According to Instance-Based Learning Theory (IBLT; Gonzalez, 2013; Gonzalez, Lerch, & Lebiere, 2003) human decisions from experience are generated through the aggregate retrieval across past experiences, based on the feature similarity of the current situation to past situations.

While IBL models provide evidence for the underlying mechanisms involved in decision making, as well as evidence that the representations used in a specific model sufficiently describe its respective task, they do not provide any insight into the degree to which a decision maker weighs particular features of the decision. We believe that thwarting a would-be attacker could be more successful if we had insight into the features they find salient.

We consider the salient feature of a decision to be the feature that most influenced that decision and the greater degree of salience a feature has, the more influential it was in the decision. While the term, salience, might imply attention, in our use the salient feature may not be the most attended feature by some measure of attention (eye gaze, etc.). It may be the case that a feature is attended more than others but ultimately does not contribute to a decision.

Our salience mechanism is somewhat analogous to gradient-based saliency used in image classification (Grün et al., 2016). Gradient-based salience techniques calculate the gradient of a prediction with respect to the input image to estimate the importance of pixels. The result of this process is often a heatmap of pixels that, when overlaid on the original image, provide some insight into what parts of the image were most important for the classification.

We term our approach *cognitive salience* in contrast to the gradient-based approach. We use the term *cognitive* for two reasons. First, our approach calculates salience by taking the derivative of a theorized memory retrieval mechanism, blending, the mechanism underlying decision making in IBL models (Lebiere, 1999). Second, the features of a cognitive model are typically of a higher-level of abstraction than pixels, usually conceptual terms, which is typical of a cognitive-level description.

In the present work, we examine the saliency of features in a model of human decisions in a cybersecurity game called the Insider Attack Game (IAG). IBL models of human decisions in the IAG revealed cognitive biases, such as confirmation bias, that emerge naturally through memory retrieval processes, and lead participants to attack far more often than predicted by perfect rationality (Cranford et al., 2020a). While much has been learned by comparing model performance to humans and making inferences about human

behavior based on the model mechanisms and processes, examining feature salience can provide further useful information regarding the relative importance of features when making decisions. These insights could prove useful in further informing how human decisions are shaped through their unique experiences, how representation of features impact decisions, and also for designing more effect cybersecurity defenses.

In what follows, we first describe the IAG and an IBL model that accurately captures human behavior in the game. Next, we describe the method for deriving cognitive salience from IBL model decisions. Assuming the model accurately reflects human decision-making processes, we apply our salience technique to the model to gain insight into how humans might weigh features when making decisions in the IAG. Finally, the results are discussed regarding their implications for theories of human decision-making and applications to cybersecurity.

IBL model of Attackers in the Insider Attack Game

The Insider Attack Game (IAG) was designed as a two-stage Stackelberg Security Game (SSG) to investigate the influence of deceptive signals on cyber-attacker decision making (Cranford et al., 2018). Players take the role of an insider attacker and make repeated decisions of "hacking" computers. In the first stage, attackers must decide which of six targets to attack, as depicted in Figure 1A. However, they must avoid the two analysts (defenders) that monitor one target each. An example target is shown in the zoomed inset of Figure 1A. Attackers are presented all information about the reward received if they successfully attack a target that is not monitored, the penalty received if they attack a target that is monitored, and the probability that the target is being monitored. After selecting a target, in the second stage, the attacker is presented with a message signaling whether the computer is being monitored (e.g., see Figure 1B). The message is always truthful when claiming a target is not being monitored. However, the attacker is informed that the message is sometimes deceptive when claiming the target is being monitored. The attacker must decide to either continue the attack and earn the reward or penalty, depending on the true underlying coverage, or withdraw and earn zero points. Attackers are incentivized to earn as many points as possible across four rounds of 25 trials each; a new set of targets are presented each round.

The defense algorithm in the IAG, the Strong Stackelberg Equilibrium with Persuasion (*peSSE*; Tambe, 2011; Xu et al., 2015), was designed to optimize the rate at which deceptive messages are sent so that belief in the signal is maintained, but does so under assumptions that adversaries make perfectly rational decisions. In the first stage, the algorithm optimizes the allocation of the two defenders based on the reward and penalty values of the targets. The algorithm effectively equalizes the expected value of all targets so that no target is more preferred than another and assigns defenders to targets across the trials according to the derived probabilities (these are the monitoring probabilities attackers see). In the second stage, the algorithm optimizes the rate of deceptive signals. If signals were only truthful, the expected value would be negative when indicating that a target is being monitored. The *peSSE* determines the probability with which to send deceptive signals so that the expected value given a signal increases to zero, and under assumptions of perfect rationality, an adversary will still defer to the safe option and withdraw. In the IAG, with a 1:3 defender-target ratio, the signal is present and truthful on 1/3rd of trials, on average. Therefore, the *peSSE* can send deceptive signals on another 1/3rd of trials when the target is not monitored. Thus, the peSSE increases the perceived coverage of the system by finding the optimal combination of bluffing (sending a deceptive message that the target is monitored when it is not) and truth-telling (sending a truthful message that the target is covered) so that a rational attacker would always withdraw in the presence of a signal.

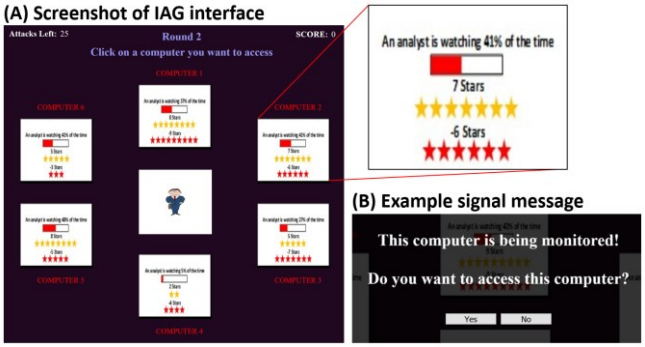


Figure 1: Screenshot of the IAG (A) and an example signal message (B). The first line of the message is omitted if the signal is absent. The zoomed inset shows a target, including the value of the reward if the attack is successful (yellow stars), the value of the penalty if the attack is unsuccessful (red stars), and the monitoring probability (given as percentage in text and graphically as a fillable gauge).

Humans, however, do not always make rational-best decisions. Instead, human decisions in the IAG can be explained under Instance-Based Learning Theory (IBLT; Gonzalez, 2013; Gonzalez et al., 2003), and an IBL model was created that captures this behavior (Cranford et al., 2018; 2020a). According to IBLT, decisions are made by generalizing across past experiences, or instances, that are similar to the current situation. In the IBL model, instances are represented by the contextual features of the decision. For example, in Figure 2, the features include the information available in the environment: the *reward*, *penalty*, *monitoring probability*, and *signal*, the *action* taken, and its associated utility, or *outcome*. Each experience is saved in memory and when a new decision is to be made, an expected outcome is retrieved from memory that represents a weighted average across all memories based on their probability of retrieval.

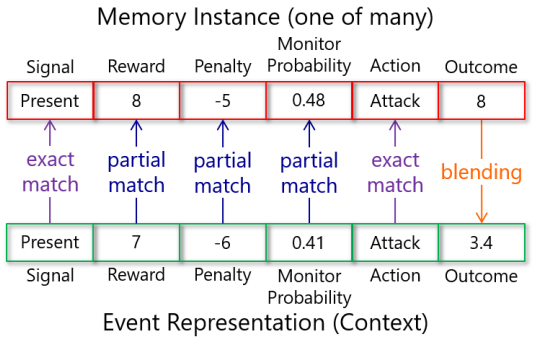


Figure 2: Example representation of instances in IBL.

The IBL model was created in the ACT-R cognitive architecture (Anderson & Lebiere, 1998; Anderson et al., 2004), which provides a theoretical framework that accurately simulates human-like cognition and processes such as memory retrieval, pattern matching, and decision making. In ACT-R, the probability of retrieving an instance is based on its activation strength which is determined by its recency and frequency of occurrence, and its similarity to the current context. The IBL model uses ACT-R's blending mechanism (described in more detail in the next section; Gonzalez et al., 2003; Lebiere, 1999) to retrieve an expected outcome of attacking a target based on a consensus of past instances. The expected outcome is the value that best satisfies the constraints of all matching instances weighted by their probability of retrieval.

The IBL model played the same game as humans. In the first stage of the IAG, the features of the decision include the monitoring probability [0.0, 1.0], the reward [1, 10], and the penalty [-1, -10]. The model generates an expected outcome for each target, via blending across previous outcomes, and selects the target with the highest expected outcome. In the second stage, the only feature in the decision is the signal [present, absent] and the model generates a new expected outcome of attacking. A straightforward decision rule is then applied: if the value is greater than zero the model attacks, else it withdraws, and ground truth feedback is given.

The model saves two instances to memory each trial. One represents the expectation generated during the decision to continue the attack or withdraw (includes the features: signal, action, and expected outcome), and the other represents the ground truth decision and feedback received (includes all features: signal, reward, penalty, monitoring probability, ground truth action, and ground truth outcome). Storing the expectations as well as the ground truth drives a confirmation bias in which the availability of additional positive instances in memory (i.e., from the positive expectations generated prior to deciding to continue an attack) perpetuates a behavior to attack when faced with a signal, even after suffering losses.

Cranford et al. (2020a) showed that humans attacked about 80% of trials on average, far more than the predicted 33% of perfectly rational attackers (i.e., on average, signals are absent on only 1/3rd of trials). The IBL model very accurately captures this behavior across trials in the game (overall total RMSE = 0.04 and r = 0.73), as shown in Figure 3 (adapted from Cranford et al., 2020a). The pattern of spikes across trials can be attributed to the schedule of coverage and signaling, which was the same for each player and reflects experiences of success/failure given the probability of seeing a signal. In fact, the correlation between the probability of seeing a warning and the probability of attacking is -0.84 for humans and -0.89 for the model.

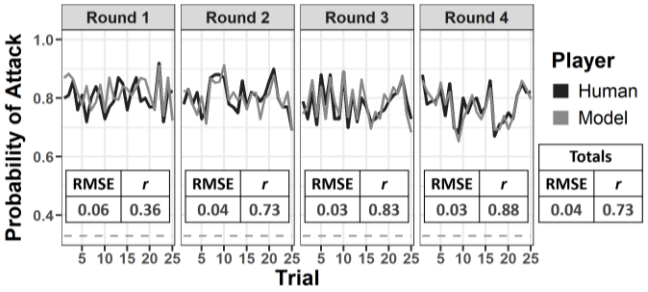


Figure 3. Mean probability of attack across trials for human participants compared to the IBL model runs.

The data presented in Figure 3 averages across substantial individual differences in behavior. In addition to capturing mean human performance across trials, the model also captures the full distribution of attack probabilities as seen in Figure 4 (adapted from Cranford et al, 2020a). Like humans, approximately 40% of participants (e.g., model runs) attack greater than or equal to 95% of trials. In another study that examined human behavior in the IAG, Cranford et al. (2020b) reported that approximately 23% of participants that attacked greater than or equal to 95% of trials also reported that they ignored the signal in their decisions. The study reported in Cranford et al. (2020a) did not collect such data, but we can assume similar responses would have been made.

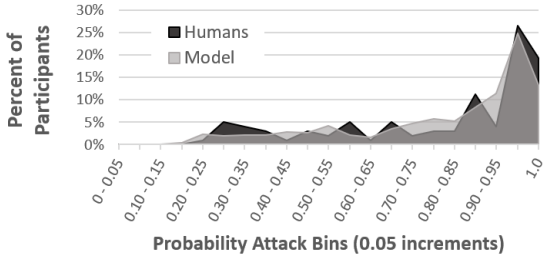


Figure 4: Probability of attack distribution for human participants compared to the IBL model runs.

In summary, the IBL model very accurately captures human behavior in the IAG and has proven useful in making inferences about human decision making in the task. Humans do not compute all information and make rational-best decisions, but instead make decisions based on past experiences, which are represented by the important features of the situation. These decisions are heavily influenced by the dynamics of memory retrieval processes which result in emergent cognitive biases (e.g., recency, frequency, and confirmation bias). These biases lead to overweighting of certain outcomes that, often, results in inflated expectations.

Humans fail to fully comply with the signal because they are more likely to expect a positive outcome than a negative one as belief in the signal deteriorates. While much has been learned about how experience influences decisions in the IAG regarding recency and frequency of instances, it is less clear how humans weigh the features in their decisions. Therefore, in the present study, we examine the salience of the features during the selection and attack decisions of the model to inform why certain decisions are made and if there are differences between types of participants in how information is processed that leads to the observed individual differences in attack behavior, as described in Figure 4.

Blending and Cognitive Salience

The cognitive salience of a feature can be derived from the blended retrieval mechanism. The blending mechanism in ACT-R retrieves an estimated outcome by interpolating across previously experienced outcomes (Lebiere, 1999). That interpolation process is weighted by the contextual similarity of the present instance to previous instances and is computed with the following equation:

$$V = \ argmin \sum_{i=1}^{n} P_i \cdot Sim(V_t, v_{it})^2$$

The value, V, therefore is an interpolated value based on matching chunks i, weighted by their retrieval probability P_i. The complete blending process is outlined in Figure 5. The retrieval probability, equation 2, is derived from a Boltzmann softmax function that is based on the activation strength of chunks, which is influenced by power laws of frequency and recency, according to ACT-R theory of memory retrieval (Anderson & Lebiere, 1998; Anderson et al., 2004), and also the similarity or match between the current instance in memory and past instances. The match score in equation 1 is equivalent to the similarity function, $Sim(V_t, v_{it})^2$, and is used to compare memory chunks v_{it} and candidate consensus values V_t. In the simplest case, where the values are numerical (i.e. the return R_T) and the similarity function is linear, the process simplifies to a weighted average by the probability of retrieval, as shown in equation 3 of Figure 5.

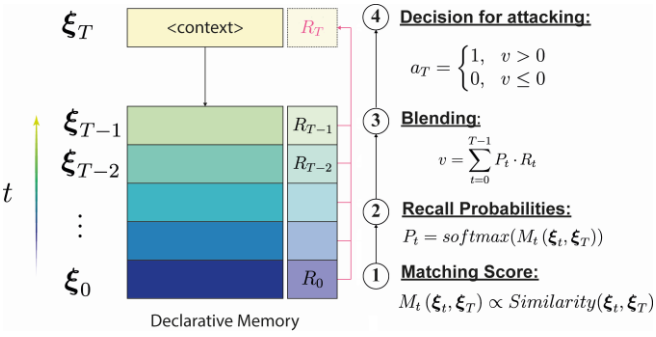


Figure 5: Description of blending mechanism.

We calculate salience by taking the derivative of the blending equation with respect to each feature:

$$S(V_t, f_k) = c \sum_{i=1}^{n} P_i \cdot \left(\frac{\partial Sim(f_k, v_{ik})}{\partial f_k} - \sum_{j=1}^{n} P_j \frac{\partial Sim(f_k, v_{jk})}{\partial f_k} \right) \cdot v_{it}$$

This derivative gives us the degree of influence a particular feature (f_k) had in a decision (V_t). The value S can be infinitely positive or negative. While the direction of the value provides information about how the feature is used, to compare between features, the magnitude tells us which feature has the greater impact on the decision. Therefore, in all analyses below, we examine the absolute values of *S*.

Cognitive salience was first applied in an explainable artificial intelligence context (Somers et al., 2019), where ACT-R was used to model a reinforcement learner (RL). In that context, the baseline equations in ACT-R, which are responsible for effects of recency and frequency, were not used because in an RL context, there is no reason to expect decay in memory. This is the first time that cognitive salience has been applied in a human experimental context.

Salience Analysis

We examined the salience of features during the first-stage, selection decision and during the second-stage, attack decision of the IBL model. During target selection, a perfectly rational attacker would display no preference for features because all targets have the same expected value. No one feature is more informative than another and do not differentiate targets. However, it is possible that, for boundedly rational humans that derive expected outcomes from very limited experiences, selection preferences could emerge if one feature becomes more salient than another. It is hypothesized that saliencies will be higher when few instances are available in memory, thus skewing the mental representation of expected values. With more experiences, the attacker should gradually and implicitly learn that all targets are of about equal values. Saliencies can inform us whether decisions reflect the statistics of the environment, that the features are by all accounts meaningless.

During the second-stage, attack decision, there are clear individual differences in the probability of attack. It is therefore hypothesized that, when the signal is present, salience for the signal is lower for those participants with high probabilities of attack. If the salience of the signal is low then, when a signal is present, the expected outcome generated should be higher because the decision will discriminate less between signal types, giving more weight to instances when a signal was not present and that have positive outcomes, thus driving up the blended outcome.

First-Stage Decision: Selection

Figure 6 shows the average magnitude (i.e., absolute value) of saliencies for the reward, penalty, and monitoring probability (Mprob) features of the selected target across the four rounds of the experiment. The saliencies presented in Figure 6 are calculated during the outcome generation

process. When generating expected outcomes, the model interpolates from previous experiences, weighing those experiences by their similarity to the features of the current target. The saliencies indicate that the model initially displays differences between features, but quickly merge within a few trials. After merging, all saliencies start out relatively high and decrease over time. The dashed gray line in Figure 6 shows the mean expected outcome across trials on the secondary y-axis. The expected outcomes are initially inflated and gradually decrease over time along with saliencies. These results suggest that the model is learning that the targets have equal expected outcomes and no feature is more salient than another. The decreasing trend in saliencies implies that the features are less influential in the decision over time. The model does not have any explicit awareness or explicit modeling of this decrease.

While the average magnitude of saliencies indicates no preferences for any particular feature, Figure 7 examines the individual differences between model runs (i.e., players) regarding the relative magnitude of the saliencies. The ternary plot takes the overall mean magnitude saliencies for each feature for each player and plots a point for each player to show the relative importance of each feature (i.e., the ratio between the three saliencies). The results show that players mostly display no preference for features as they learn over time that no feature is more meaningful and the saliencies trend downwards (as shown in Figure 6). Figure 7 shows that players mostly center around the middle point of the plot, indicating saliencies for features are all close to the same magnitude. However, all players display a higher saliency for one of the features, even if miniscule, and some do extend toward the corners of the ternary plot if only by a small percentage. Players were therefore split into 3 groups depending on the feature that is overall most salient, the reward, penalty, or Mprob, to examine if these groups display any target selection preferences.

Figure 8 shows a scatterplot of target selections by the reward and penalty values of the selected target. The size of the dots indicates the percent of selections for that target within a round. Thus, the dominant color for a point indicates a greater percent of selections for that group on that target. The distributions of penalties and rewards for each "Max Saliency" group are shown as marginal plots on the right and top, respectively. Figure 8 shows clear differences in target selection behavior between groups. The players that have higher saliency for the penalty tend to select the targets with higher penalties, which incidentally also have higher rewards. Meanwhile, players that have higher saliency for reward or Mprob tend to select the targets with lower penalties, and the ones with more moderate reward/penalty tradeoffs, more often.

Second-Stage Decision: Attack

In the second-stage decision, it was hypothesized that *aggressive* attackers (i.e., those that attacked ≥ 95% of the time) may weigh the signal differently than the other *cautious* attackers. Figure 9 shows the mean magnitude of saliencies

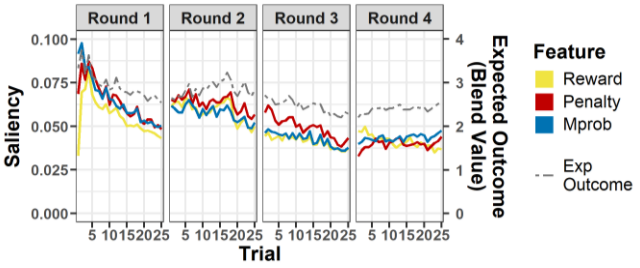


Figure 6: Mean magnitude of saliencies across trials for each feature of the selected targeted. The dashed gray line shows the mean expected outcome of the selected target.

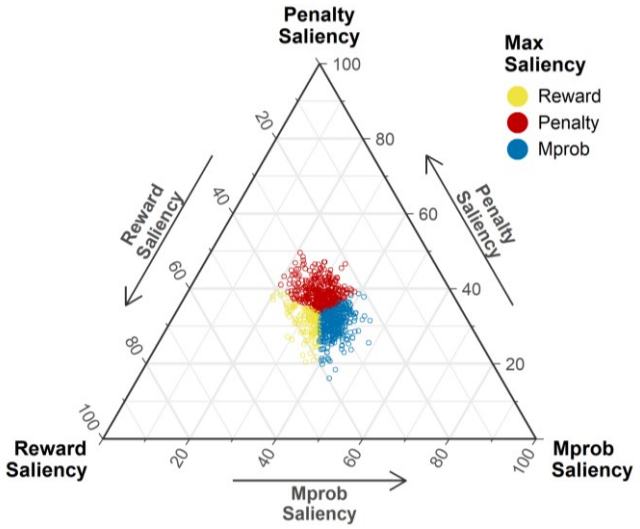


Figure 7: Mean relative saliency per player. Colors indicate the most salient feature for that player.

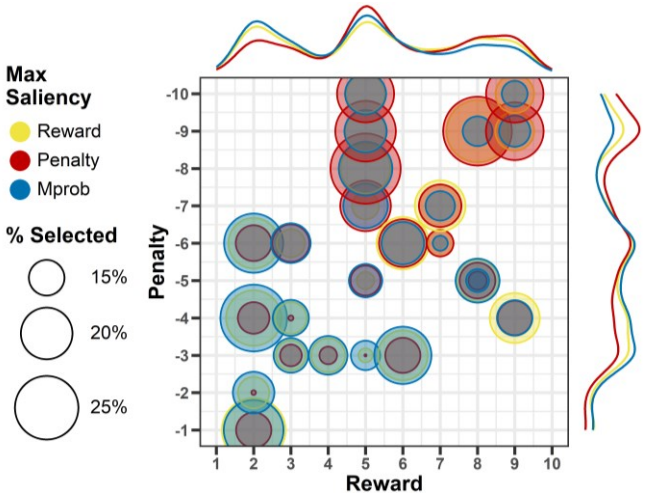


Figure 8: Scatterplot of reward and penalty of selected targets for each Max Saliency group. The size of the bubbles represent the percent of selections for the target within each Max Saliency group and round.

for the signal feature across the four rounds. The breaks in the solid line for the signal-absent condition are trials in which every target was scheduled to present a signal. When the signal is absent, the saliency for the signal is about equal in Round 1, but is higher for the aggressive attackers through the remainder of the game. For both player types, the saliency is overall higher when the signal is absent than when the signal is present. When the signal is present, however, the salience of the signal is lower for aggressive attackers than for the cautious ones. As predicted, when present, the signal is less influential in aggressive attackers' decisions than cautious attackers' decisions.

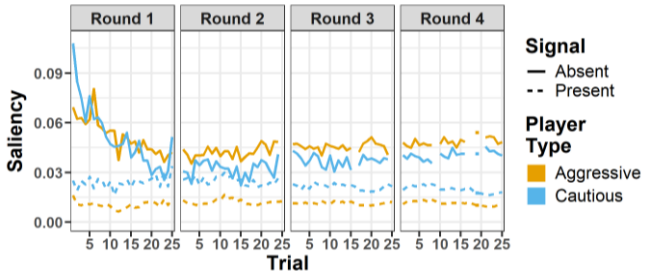


Figure 9: Mean magnitude of salience for the signal during the attack decision, for each signal and player type.

Figure 10 shows the mean expected outcome for the attack decision across the four rounds. An interesting interaction presents when compared to the pattern for saliencies. For expected outcomes, when the signal is absent, the aggressive attackers generate higher values than cautious attackers, echoing the pattern for saliencies. However, in contrast to saliencies, when the signal is present, cautious attackers generate lower expected outcomes than aggressive attackers.

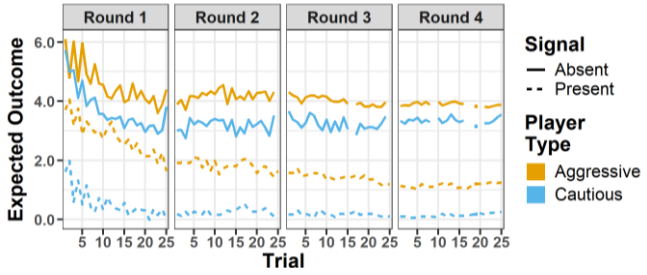


Figure 10: Mean expected outcome of the attack decision, for each signal and player type

These results suggest that salience influences decisions through its influence on memory retrieval. A higher saliency for a signal means that, for blending, more weight is given to past instances that have the same value as the current signal, and when saliency is low for the signal more weight is given to past instances that have a different value as the current signal so that probabilities are more evenly distributed across past instances. When a signal is absent, the higher salience for aggressive attackers means more weight is given to past instances whose signal is absent. Because these instances have only positive values, the expected outcomes are inflated. When a signal is present, the lower salience for aggressive attackers means the probability of retrieving past instances whose signal is absent is higher, and more evenly distributed across all targets (hence the expected outcome by Round 4 is approximately equal to the true expected value, irrespective of the signal, of 1.43), and again the expected outcomes are inflated. Meanwhile, cautious attackers that have higher salience for the signal when present have expected outcomes near zero, which is the true expected value given a signal.

Conclusion

The present study is the first to use this method for calculating cognitive salience to introspect into the model how humans weigh the contextual features in their decisions. The results provide additional insight into how the representation of features can influence decisions. Specifically, we can infer that players learn quickly that all targets have equal expected values and no feature is more informative than another. But also, because human decisions are based on limited experiences, some individuals have a slight preference for some targets that can be predicted by the degree of salience for a particular feature. An interesting path for future research will be to examine whether target selection preferences emerge if the targets are not of equal expected value and/or certain features are more indicative of success than others, as was shown in the original work on cognitive salience in an explainable artificial intelligence context (Somers et al., 2019). In that research, feature preferences emerge as certain features are more indicative of successful decisions.

In contrast to the selection decision, during the attack decision the degree of salience for the signal has a direct impact on the probability of retrieving past instances which in turn impacts the expected outcomes generated. The results provide unique insights into how individual differences can emerge through unique experiences. Understanding how an individual weighs the feature in their decisions provides valuable evidence about how information is processed and how it impacts decisions, which is vitally important for improving security defenses, especially for defenses that rely on adaptive and personalized defense (Cranford et al., 2020b). Therefore, one avenue for future research will be to validate the observed model results with human experiments designed to investigate what features are most important in decisions. For example, Cranford et al. (2020b) showed that aggressive attackers tended to report that they ignored the signal feature, and a model that omitted that feature from the representation was a better predictor of these aggressive attackers. As was demonstrated, understanding what features are important in a decision can inform the design of models and about the underlying representation of the decision. More accurate models can provide more accurate predictions about human behavior which can be used to improve security algorithms. Examining cognitive salience with cognitive models provides valuable information about individual differences between players, and future research aims at exploring its utility for designing more effective personalized, adaptive signaling schemes for cyber defense.

Acknowledgments

This research was funded by DARPA XAI contract FA8650-17-C-7710 and by the Army Research Office MURI Grant Number W911NF-17-1-0370.

References

Anderson, J. R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Mahwah, NJ: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036-1060.

Cranford, E. A., Lebiere, C., Gonzalez, C., Cooney, S., Vayanos, P., & Tambe, M. (2018). Learning about cyber deception through simulations: Predictions of human decision making with deceptive signals in Stackelberg Security Games. In *Proceedings of the 40th annual conference of the Cognitive Science Society* (pp.258-263). Madison, WI: Cognitive Science Society.

Cranford, E. A., Gonzalez, C., Aggarwal, P., Cooney, S., Tambe, M., & Lebiere, C. (2020a). Towards personalized deceptive signaling for cyber defense using cognitive models. *Topics in Cognitive Science* (under review).

Cranford, E. A., Gonzalez, C., Aggarwal, P., Cooney, S., Tambe, M., & Lebiere, C. (2020b). Adaptive cyber deception: Cognitively-informed signaling for cyber defense. In *Proceedings of the 53rd Hawaii International Conference on System Sciences* (pp. 1885-1894). Maui, HI.

Gonzalez, C. (2013). The boundaries of instance-based learning theory for explaining decisions from experience. *Progress in Brain Research*, 202, 73-98.

Gonzalez, C., & Dutt, V. (2011). Instance-based learning: Integrating decisions from experience in sampling and repeated choice paradigms. *Psychological Review, 118*(4), 523-551.

Gonzalez, C., & Lebiere, C. (2005). Instance-based cognitive models of decision making. In D. Zizzo & A. Courakis (Eds.), *Transfer of knowledge in economic decision-making*. Macmillan (Palgrave Macmillan).

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance based learning in dynamic decision making. *Cognitive Science*, *27*(4), 591-635.

Grün, F., Rupprecht, C., Navab, N., & Tombari, F. (2016). A taxonomy and library for visualizing learned features in convolutional neural networks. arXiv preprint arXiv:1606.07757.

Lebiere, C. (1999). A blending process for aggregate retrievals. In *Proceedings of the 6th ACT-R Workshop*. George Mason University, Fairfax, Va.

Lebiere, C., Wallach, D., & West, R. L. (2000). A memory-based account of the prisoner's dilemma and other 2x2 games. *Proceedings of International Conference on Cognitive Modeling* (pp. 185-193). NL: Universal Press.

Sanner, S., Anderson, J. R., Lebiere, C., & Lovett, M. C. (2000). Achieving efficient and cognitively plausible learning in Backgammon. *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

Somers, S., Mitsopoulos, K., Lebiere, C., & Thomson, R. (2019). Cognitive-Level Salience for Explainable Artificial Intelligence. In *Proceedings of the 17th Annual Meeting of the International Conference on Cognitive Modeling*. Montreal, CA.

Tambe, M. (2011). Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned. *Cambridge University Press.*

West, R. L., & Lebiere, C. (2001). Simple games as dynamic, coupled systems: Randomness and other emergent properties. *Journal of Cognitive Systems Research, 1(4)*, 221-239.

Xu, H., Rabinovich, Z., Dughmi, S., & Tambe, M. (2015). Exploring information asymmetry in two-stage security games. In *Proceedings of the National Conference on Artificial Intelligence* (2, pp. 1057-1063). Austin, TX: Elsevier B.V.

Missed One! How Ballot Layout and Visual Task Strategy Can Interact to Produce Voting Errors

Joshua Engels[1], Xianni Wang[2], Michael D. Byrne[1,2]
{jae4, xw48, byrne}@rice.edu
[1]Department of Computer Science, [2]Department of Psychological Sciences
6100 Main St., MS-25, Houston, TX 77005 USA

Abstract

This paper presents an ACT-R model designed to simulate voting behavior on full-face paper ballots. The model implements a non-standard voting strategy: the strategy votes first from left to right on a ballot and then from top to bottom. We ran this model on 6600 randomly-generated ballots governed by three different variables that affected the visual layout of the ballot. The findings suggest that our model's error behavior is emergent and sensitive to ballot structure. These results represent an important step towards our goal of creating a software tool capable of identifying bad ballot design.

Keywords: ACT-R; error prediction; voting

Introduction

Voting is hard. The deliberations and conversations that go into choosing who best represents one's interests is an important and time-consuming task, one that might be argued to be the very backbone of a democracy. Understandably, many may believe that the subsequent task of correctly indicating one's chosen candidate is comparatively easy and straightforward. Surely once a voter gets the ballot and can mark whoever they please, the hard part is over.

Often, this is correct. When ballots are designed well, errors voters make are not systematic and generally will not help or hurt any particular candidate. However, when ballots are designed poorly, they may lead to systematic voting errors. It is possible such errors do not matter if margins of victory are large and thus such issues may go unnoticed.

However, in closely-contested elections it is not the general case that is important. There have been numerous elections in the past 20 years that have been documented as having been decided by systematic voting errors caused by bad ballot design. This ranges from the infamous "butterfly ballot" in Palm Beach County, Florida in the year 2000 (Wand, et al., 2001) to the most recent major U.S. election in 2018, where a U.S. Senate seat (also in Florida) was almost certainly decided by a poorly-designed ballot (Chisnell & Quesenbery, 2018). For a review, see Norden, et al. (2008).

While election interference by hacking is a far more flashy and obvious risk, there has never been clear evidence that this has swung an election, unlike with bad ballot design. Ironically, the fear of hacking has led to a return to paper ballots, which with their profusion of races packed onto small sheets of paper makes ballot design even more important.

The most likely errors caused by poor ballot design are under- and overvoting. An undervote is an error that occurs when the voter fails to vote on a race that they intended to, whereas an overvote is when a voter votes on a race more than the allowable number of times (usually, more than once). The problem of designing a ballot that will not cause people to systematically under- or overvote is challenging. For instance, it might entail running a usability study weeks before the actual election. What makes the problem so difficult is the sheer number of counties in the United States (over 3000), each of which designs their ballots differently and each of which have hundreds of different iterations of ballots for each precinct they are responsible for. Manually checking each ballot with a usability study is infeasible.

One possible solution to this problem is software that could automatically check an arbitrary ballot for common design errors. However, such a solution would only find errors that had been previously made by voters on other ballots. If the task is to predict if humans will make a mistake on a novel ballot, it is difficult to imagine that chasing only known errors will be sufficient. Here is where ACT-R (Anderson, 2007) modeling comes in. Since ACT-R is generative, it can predict behavior on any ballot and is not limited only to errors that have been made previously.

Building such a predictive model is itself an extremely challenging task because it would have to be able to predict all historical voting errors as well as any new ones. For example, while Green (2010) built an ACT-R model that could make the same mistake voters did in a specific famous ballot (the 2006 Sarasota County ballot), it was limited to replicating one specific error behavior.

Thus, Wang, Lindstedt, and Byrne (2019) present the outline for an ambitious project: a model that can simulate the entire space of possible voting behaviors. They presented a smaller scale version of this end goal model. The model ran in a voting environment called VoteBox, a simulated electronic ballot, consisting of a single race per screen and a "next" button to navigate.

Nevertheless, within just this simple task was hidden great complexity: the model used a total of 40 different voting strategies constructed from differing levels of ballot/candidate knowledge and navigational strategy selections. The voters differing strategies and knowledge led to different rates of error, showing that a model voter's strategy made a difference on whether or not it was able to accurately vote for its intended candidates. However, this effort was preliminary in that it did not vary the design of the ballot; it simply demonstrated that errors were emergent from a particular combination of task strategy and memory contents.

In this paper, we describe a model that represents the natural extension of this system to show that errors can emerge from the interaction of strategy and ballot design. This model also works in a more challenging visual environment: it handles simulated full-face paper ballots. A full-face ballot is one that has all the races on a single display (usually a piece of paper). This extension introduces new model building challenges. Our new models must navigate both between and within races, and our model creation process must be flexible enough to explore an even larger voting strategy space.

Unsurprisingly, the increased complexity of a full-face paper ballot leads to increased model error. Thus, we also describe the error rates of simulated voters on differing simulated ballots. This represents an important step towards our end goal of constructing a generative model able to identify bad ballots.

Method

First, we will describe the design of our full-face ballots, then the design of the model, and our simulation of the model across many possible ballot designs.

Ballot Design

We built simulated full-face paper ballots for the model which consist of a virtual screen populated with several columns of races. Each race has a title, a list of candidates and their associated parties, and a list of buttons that the model can click to vote for a candidate. (see Figure 1).

Figure 1: Top left corner of a simulated ballot.

The resulting simulation is not quite the same as an actual paper ballot. For example, the model clicks on a button instead of filling in a circle and does not obscure the ballot with its hand while doing so. However, the ballot is typical in visual layout, which we believe is similar enough to cause many of the same errors we expect human voters to make.

Because ACT-R's nascent ability to group visual items is somewhat limited (Lindstedt & Byrne, 2018), we had to work around this. So, to help the model navigate, we colored the race header red, the candidates purple, and the parties blue. The coloring allows the model to make visual location requests like "the closest red text in the column to the right" (when finding the closest race) or "the closest purple text to my current position" (when finding the candidate group of the currently attended race). Since we suspect humans can also reliably differentiate between race headers, candidates, and parties by using the visual characteristics of the ballot, we believe coloring the ballot does not give the model an unfair advantage. However, we are exploring alternative ways to work around this problem.

Model Design

We built the model with one overarching goal in mind: to simulate as wide an array of voters as possible.

Our modular system split a simulated voter's strategy into four different pieces: (1) macronavigation, the process of moving from one race to the next; (2) visual encoding, the process of determining the race, party, and candidate visual groups for each race; (3) micronavigation, the process of finding the intended candidate to vote for within each race; and (4) selection, the process of actually clicking on the button corresponding to the chosen candidate. At runtime we selected one strategy from each of these categories and combined them together with a declarative memory file to build an ACT-R model. Note that how the model does pieces 2–4 was taken directly from the Wang et al. (2019) model.

Designing A New Strategy

We first built the most obvious option for each strategy category because we wanted our initial strategies to lead to a composite voting strategy with no errors. We wanted to ensure that our model worked before we started varying pieces to induce errors.

Our first strategy after these obvious ones was a non-standard macronavigation strategy. Our model's standard macronavigation strategy was *top to bottom left to right*; that is, the model started in the top left corner and went all the way to the bottom of the column and then went over to the next column to the right and again went top-to-bottom, repeating until it was finished. This is the most obvious method of macronavigation, and as noted above resulted in no mistaken votes. The first alternative macronavigation strategy we built was *left to right top to bottom*.

The *left to right top to bottom* strategy starts on the upper leftmost race on the ballot. It then proceeds to the right, navigating to the closest race to the last race it voted on in the next column over, and repeating until it votes on a race in the last column. Then, it goes back to the beginning of the row, finds the next race down in the left column, and repeats voting from left to right. The model continues until it runs out of new races in the left column.

On our ballots the races in each column are horizontally aligned, as might be expected. However, when race lengths

Figure 2: The green arrows mark the first part of the *left to right top to bottom* model's voting pattern on this specific ballot. The model skips CommisionerofAgriculture.

are allowed to vary, races in different columns are not *vertically* aligned, as the generation process always placed each race a set distance below the last race. Because our new macronavigation strategy proceeded initially from left to right, when races were vertically misaligned our model *could miss races*. Note that when the ballot is a perfect grid where all races are vertically aligned, the model does *not* make errors. It is the interaction of this strategy with the design of the ballot that results in errors. For an example of the model missing a race on a typical ballot, see Figure 2.

In Figure 2, when the model reaches the third race down in the left column ("United States Representative District 7") it votes on that race and then proceeds right along the row, selecting and voting on the closest race and repeating until it reaches the last column. The model then returns to the race at the beginning of the row and proceeds to the first race on the next row down ("Governor"). Here is where it makes its mistake: because the "Railroad Commissioner" race is the closest race to "Governor," the model votes on "Railroad Commissioner" for its second race in the row and so skips Commissioner of Agriculture. It never returns and votes on this race.

We observed that the races our new strategy missed depended on the layout of the races on the ballot and determined it was critical to understand if this was systematic.

Experiments

Once we had a simulated voter making structure-based mistakes, we decided to test how these mistakes changed as a function of the ballot layout. Initially, our ballot was static, consisting of a manually-positioned set of races and candidates. Our first step was modifying the ballot so it could be dynamically generated. Every time we ran the model, our generation process allowed us to vary the vertical spacing between races, the vertical space between the race header and the candidates, and the vertical space between candidates. We chose ranges of the variables that led to ballots our model could still realistically parse but that nevertheless were visually distinct (see Table 1). As the ballot was generated each race was randomly selected to have between 1 and 4 candidates.

Table 1: Ballot Layout Variables

Variable	Range (Pixels)
Space between races	5 – 15
Space between header and candidates	20 - 22
Space between candidates	15 - 18

For each one of the 132 possible combinations of spacing variables (see Table 1), we ran the model on 50 randomly generated ballots. Thus, our model was run on 6,600 ballots for a total count of 158,338 individual races. For each run, we recorded the exact race positions and race order on the ballot, as well as the order the model voted on races (including any races the model missed).

The data allow us to characterize this strategy and identify how and where it fails. We will also describe good and bad ballot design by seeing which designs lead to more error in the model. This will serve as a case study for how new strategies built on our architecture will find errors in novel ballots.

Results

First, we define model *percent error*, the percent of races that our model skips. Our model's *global* percent error is around 13.04%, meaning that, on average, given a random race on a ballot there is a 13.04% chance that our model will not vote on it. This rate is certainly much higher than any experimental rate in human voters, but as this strategy is nonstandard, this result is to be expected. Of course, most people do not make anywhere near these many errors, but average error rates in the wild likely stem from outliers like this strategy.

Effects of Race Location

We first examine the relationship of race location on the ballot to model error. We observe that there is a general trend of increasing error across columns (see Figure 3). In other words, races in columns that are further to the right are more likely to be skipped.

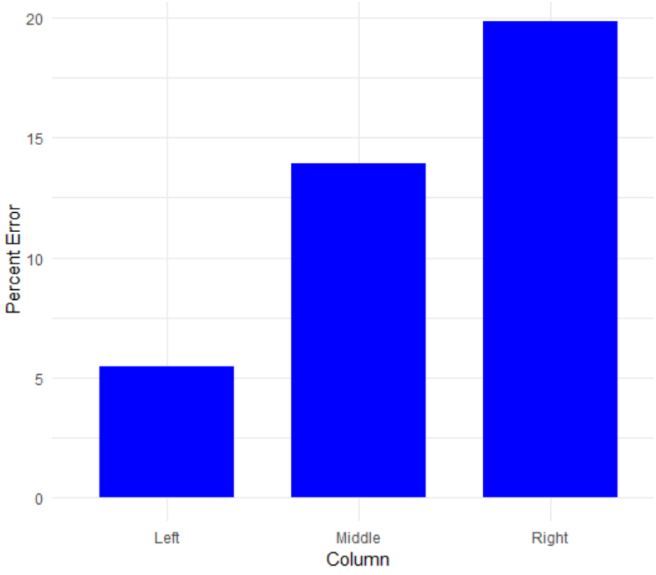


Figure 3: Average percent error across races in the left, middle, and right column across all ballot runs

In fact, since we recorded the exact *y* coordinate and column for every race on every ballot, we can generate a heatmap of error rate by race position on the ballot (see Figure 4). Each bin collates the percent error of the model on races within 10 vertical pixels, where the *y* position of a race is its header.

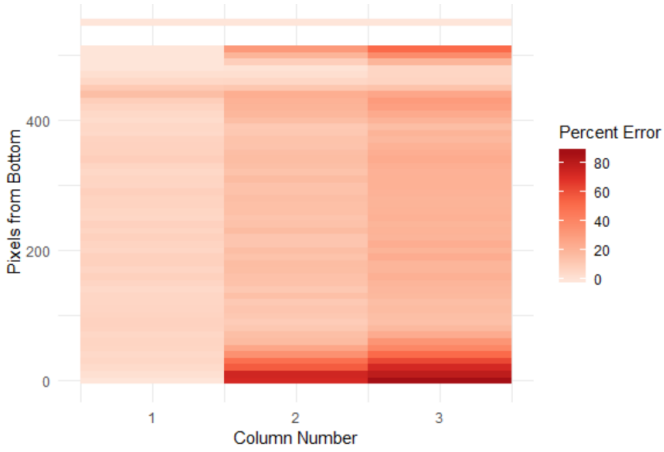


Figure 4: Heatmap of the model's error according to races' column number and *y* position.

Of interest are places in Figure 4 where errors are likely. One immediately obvious place is the bottom right corner, where average percent error approaches 1. The model almost always misses races there. To make sense of this result, we observe that the only way a race can have its *start* in one of those bottom right boxes is if it is very short. It makes sense that for short races nestled in the bottom corner, people will frequently get to the last race in the left column and vote across that row not *low enough* to reach the bottom corner races.

However, other than this, errors are more or less uniformly distributed across the ballot. This result hints at the strength of our model: errors occur seemingly randomly across the ballot because they are emerging from the specific structure of individual random ballots. Thus, using our data of each experiment's race layout, we move onto examining how specific elements of ballot structure influence model error.

Effects of Ballot Structure

We first examine the error rate as we vary the amount of vertical space between the end of each race and the beginning of the next. Recall that vertical space is just one of the spacing variables we manipulated (see Table 1). Thus, each specific vertical spacing value includes many observations from ballots built from combinations of the other spacing variables. While we did examine these other spacing variables, we found they had no significant effect on the model's error rate.

As the space between races decreased, voting error increased (see Figure 5). This result validates the intuition that the more cluttered a ballot is, the more likely a simulated voter is to miss a race.

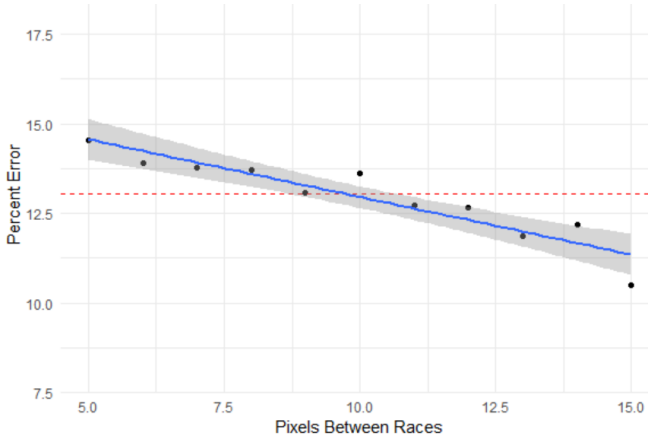


Figure 5: Each black dot is the average percent error across all ballots with a specific race spacing. The blue line is the linear regression for the trend, the red line is the average error of the model, and the shading represents 95% confidence intervals for the line.

We also examined how the length of a race was related to the chance it would be skipped and found similar results: as the length of a race decreased, the model's chance of skipping it (its error rate for races of that length) increased (see Figure 6). Of note, single-candidate races are most likely to be missed, but of course skipping such a race will not change the outcome of an election, since unopposed candidates are guaranteed to win.

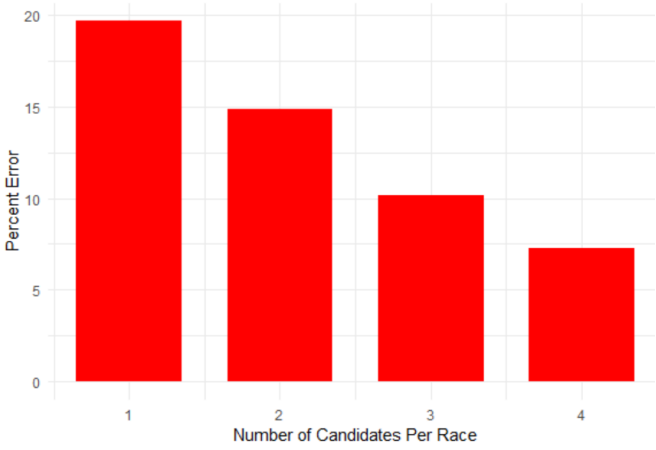


Figure 6: Average error rate of the model on races of one candidate, two candidates, three candidates, and four candidates.

Finally, we looked at how the model's error rate varied as a function of the vertical distance from a given race to the nearest race to it in the last column. In Figure 7, we show a stacked bar plot of races missed and races voting on according to this variable. This graph shows two things: one, that the chance a simulated voter misses a race increases as the closest distance to the last race increases, and two, that the number of races that are far from any prior race decreases as the distance increases. The reason that the distribution is non uniform, with peaks in the 0 bin, 15-20 bin, 30-35 bin, and 45-50 bin, is a result of the way ballots were generated. The candidate spacing varied from 15 to 18 pixels (see Figure 2), and it was frequently the case that the closest race in the last column was an integer multiple of candidate space away.

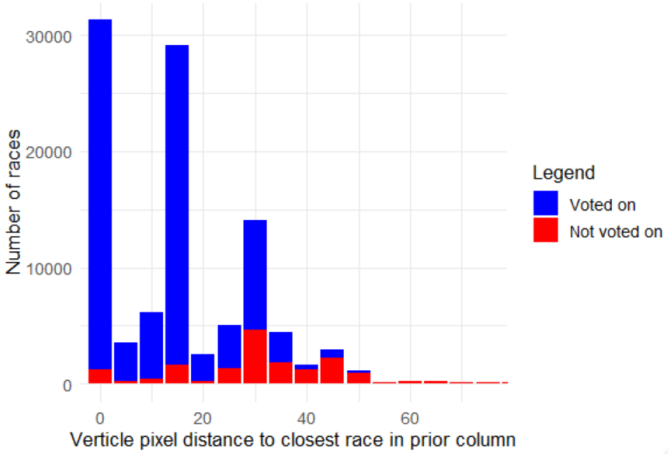


Figure 7: Stacked bar plot of the number of races voted on and not voted on across all model runs, plotted according to the vertical distance between the race and the closest race in the last column (bins of 5 pixels).

This graph more than any other illustrates the model's tendency to miss races that are not lined up in a row; building and running the simulation allows us to identify what these races are for any given ballot.

Conclusion

Races were more likely to be missed if they were smaller, out of alignment with the races in other columns, or more cramped overall. These are all characteristics of bad ballots that our model detected organically. The detection behavior emerged out of the design of the strategy; it was *not* hardcoded. The fact that the model's error behavior was unplanned and emergent is in line with the long-term plan of building models that can produce novel errors on novel ballots.

Notably, using a non-standard macronavigation strategy amplified our ability to detect bad ballots. For instance, a strategy moving in the same direction as the races were originally placed might not mind if the races were very close together, but any other strategy would. Ballot designers need to cater to less common strategies, so an ability to detect when ballots will cause systematic errors in voters using these strategies is crucial.

Indeed, we should note that the average error for this strategy is far higher than the average error for all voters, even assuming as we did that once a voter found a race they would successfully vote on it (choosing a perfect micronavigation strategy, in the parlance of our model). Most real voters probably use a more successful macronavigation

strategy. They also may take additional steps we do not yet account for, like scanning the ballot again to see if they missed any races. However, if even a subset of voters uses this strategy, or one like it, then we must account for them in our model, as a subset of voters can still have a deciding impact on a close race.

Thus, one of our next steps will be to map the space of macronavigation strategies by running eye tracking experiments. This research will seek both to find new types of voting strategies and to estimate their prevalence in the voting population. Then, once we build models that represent all of these voting strategies, we will be able to build a ballot analysis tool that runs ballots through each model and weights the resulting error rates by how often people actually use the strategy. Our goal is to be able to use this tool to come up with a global error rate prediction for an arbitrary ballot, preventing badly designed ballots from ever reaching voters.

To implement these new strategies, we will need to expand the capabilities of ACT-R itself. We plan to start by extending the visual grouping module to group objects in a hierarchy and by adding new options for visual navigation. With these new capabilities, we will be able to build new sub-strategies for the model, including new ways for the model to encode the candidate, party, and race groups and new ways for the model to find and click the circle corresponding to a candidate. Each strategy will have a characteristic error pattern like we described in this paper, and together the set of strategies will span the possible space of errors.

Thus, while some of the findings in this paper may seem obvious, they must partly be viewed in the light of the wider project. Our model was able to vote on a wide array of ballots that looked visually different and successfully make consistent errors. More than just characterizing the type of ballots and races that are more disposed to be skipped by a specific voter, these findings confirm the feasibility of attempting to eventually predict errors in novel ballots.

Furthermore, the model makes an interesting additional prediction: since our model is more likely to miss races in the center and right columns, and more likely to miss smaller races, the models predicts that average voter error should be *higher* on down ballot races in the real world (as some voters may use a similar left to right strategy). This skew is likely to be more severe in years with a presidential race, since there are often many candidates running for president, meaning that the first race in the left column would be very long, thus making it more likely that other columns races will not be aligned.

We can even use our results to generate applied advice for a hypothetical election official who must build a ballot with races of varying length. Such an official should strive to line up race headers as much as possible, sacrificing races per page by leaving blank space so that races can be aligned (this would help increase accuracy not only with the specific strategy we tested, but any strategy that goes left to right). Moreover, the official should try not to squeeze races into the bottom right corner, and in general try to keep the ballot uncluttered by putting as much space between races as possible. The official might even consider making the space *within* races more cramped to make the delineations *between* races clearer, although this will introduce the possibility for a voter filling in the wrong bubble or missing the candidate they want to vote for. Future models we build will predict these errors as we continue towards our goal of constructing a model that can simulate all possible voter behavior.

Acknowledgments

This research was supported by grants #IIS-1920513 and #CNS-1550936 from the National Science Foundation. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of NSF, the U.S. Government, or any other organization.

References

Anderson, J.R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Chisnell, D., & Quesenbery, W. (2018). How a badly-designed ballot might have swayed the election in Florida. Washington Post, November 12, 2018. Retrieved from https://www.washingtonpost.com/outlook/2018/11/12/how-badly-designed-ballot-might-have-swayed-election-florida/

Lindstedt, J. K., & Byrne, M. D. (2018). Simple agglomerative visual grouping for ACT-R. In I. Juvina, J. Houpt, & C. Myers (Eds.), *Proceedings of the 16th International Conference on Cognitive Modeling* (pp. 68–73). Madison, WI: University of Wisconsin.

Norden, L., Kimball, D., Quesenbery, W., & Chen, M. (2008). *Better Ballots*. New York, NY: Brennan Center for Justice, NYU School of Law.

Wand, J. N., Shotts, K. W., Sekhon, J. S., Mebane, W. R., Herron, M. C., & Brady, H. E. (2001). The butterfly did it: The aberrant vote for Buchanan in Palm Beach County, Florida. *American Political Science Review, 95*(4), 793–810.

Wang, X., Lindstedt, J. K., & Byrne, M. D. (2019). The model that knew too much: The interaction between strategy and memory as a source of voting error. In Stewart, T.C. (Ed.) *Proceedings of the 17th International Conference on Cognitive Modeling* (pp. 283–288).Waterloo, Canada: University of Waterloo.

Time-related Effects of Speed on Motor Skill Acquisition

Pierre G. Gianferrara (pgianfer@andrew.cmu.edu)
Shawn Betts (sabetts@andrew.cmu.edu)
John R. Anderson (ja0s@andrew.cmu.edu)
Department of Psychology, Carnegie Mellon University, 5000 Forbes Ave
Pittsburgh, PA 15213 USA

Abstract

Anderson *et al*. (2019) present an ACT-R model of how humans learn to play rapid-action video games. To further test this model, we utilized new measures of action timing and sequencing to predict skill acquisition in a controlled motor task named *Auto Orbit*. Our first goal was to use these measures to capture time-related effects of speed on motor skill acquisition, operationalized as a performance score. Our second goal was to compare human and model motor skill learning. Our results suggest that humans rely on different motor timing systems in the sub- and supra-second time scales. While our model successfully learned to play *Auto Orbit*, some discrepancies in terms of motor learning were noted as well. Future research is needed to improve the current model parameterization and enable ACT-R's motor module to engage in rhythmic behavior at fast speeds.

Keywords: timing; sequencing; motor control; speed; rhythm; variability; ACT-R; motor skill acquisition

Introduction

Many everyday tasks such as typing, chopping, and playing a musical instrument require one to acquire complex motor skills. Motor skill learning has been defined as the neuronal changes that enable an organism to execute a motor task better, faster, and more accurately over time (Diedrichsen & Kornysheva, 2015). Evidence from the motor control literature suggests that humans may rely on chunking strategies over the course of motor skill learning resulting in more predictable motor action sequences (Verstynen *et al*., 2012; Beukema, Diedrichsen & Verstynen, 2019).

In addition to the chunking of motor actions, one needs to incrementally estimate the correct timing of upcoming motor actions (Decety, Jeannerod & Prablanc, 1989; Palmer & Pfordresher, 2003). One common model of timing is the attentional gate model developed by Zakay and Block (1996) which relies on the generation of regular pulses to keep track of time. However, prior research on the neuroscience of timing provides evidence suggesting that this model may not be sufficient to capture the full complexity of motor timing (Coull, Cheng & Meck, 2011; Breska & Ivry, 2016).

According to Lewis and Miall (2003), motor timing may differ depending on the time scale of the motor actions that are executed. On the one hand, motor actions in the second range were shown to depend on a 'cognitively controlled' timing system heavily dependent on prefrontal and parietal neural processing. On the other hand, motor actions in the sub-second range were shown to depend on an 'automatic' timing system mostly dependent on motor circuits (Lewis & Miall, 2003).

In this study, we used the ACT-R production system, which successfully modeled skill acquisition in a previous complex task (Anderson *et al*., 2019), to assess potential time-related effects of speed on motor skill acquisition. To do so, we designed a novel video game, *Auto Orbit,* inspired by *Space Fortress* (Mané & Donchin, 1989). Our main goal was to determine to what extent ACT-R is currently able to capture the detail of human motor learning in such games. In this study, we manipulated the game speed, such that agents would play the same video game at a faster or slower speed. Our principal analysis compared ACT-R's motor learning to human learning and strove to capture the different elements of motor skill acquisition in a time-dependent statistical framework.

Auto Orbit Video Game

In *Auto Orbit,* a spaceship is rotating in an orbit at a fixed speed around a balloon (circle-shaped target) placed in the middle of the screen (see Figure 1). The player must periodically adjust the ship's aim and regularly shoot missiles within a specific firing interval. Each successful shot triggers a quick electronic sound and results in the balloon being inflated by 1/10 of its full size. Once the balloon is fully inflated, the player needs to execute a quick double shot shorter than 250 ms to burst the balloon and complete a game cycle. Balloon bursts were each rewarded by a fixed number of points dependent on the game speed, while misses (unsuccessful missiles) resulted in a penalty of 2 points. Each game was broken down into a series of game cycles that started with a "balloon respawn" game event and ended with a "balloon burst" game event. For each game, a log file was recorded with 16-ms temporal resolution.

Controlling the space ship in *Auto Orbit* involved three actions: rotating clockwise by 15 degrees ("D" key), rotating counterclockwise by 15 degrees ("A" key), and launching a missile ("L" key). During the game, the specified firing interval was learned through balloon resets as the lower bound, and balloon deflations as the upper bound. Each shot that was faster than the lower bound resulted in a reset characterized by the balloon popping on the screen. Conversely, the player's failure to hit the balloon before the upper bound resulted in a balloon deflation characterized by the balloon dwindling at a constant

deflation rate of 1 % of the balloon's full size or 0.18 pixels per game tick. Finally, in order to add some noise in the video game, random ship rotations of 60 to 120 degrees occurred with 1/3 probability at the beginning of every game cycle, and the players were then given additional time to re-adjust the ship's aim. Rotation onset time was randomly generated according to a uniform distribution with 1 s. as the minimum rotation onset time and 4 s. as the maximum rotation onset time. After each ship rotation, agents had 2 s. to adjust the ship's aim and continue firing before the balloon started deflating. An illustration of the *Auto Orbit* interface is depicted on Figure 1. One can play the *Auto Orbit* video game using the following link: http://andersonlab.net/orbit/signin.html

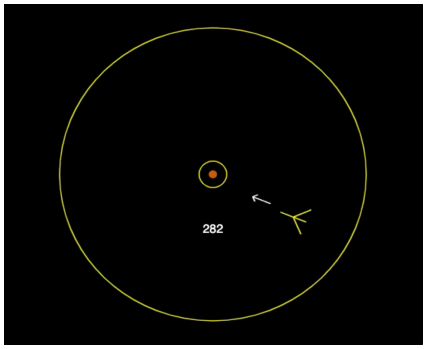


Figure 1: Visualization of the *Auto Orbit* video game interface.

ACT-R Model of Skill Acquisition

The ACT-R model was adapted from past modelling work by Anderson *et al*. (2019). First, operators were stored in declarative memory to represent the model's strategy for playing the video game. Operators were set up such that the model would first adjust the ship's aim (angular orientation relative to the balloon center) and would then monitor the timing of its shots (time that needed to elapse after a shot before the model could fire another missile). Over the games, the model became faster in executing these operators through a process called production compilation, which converts operators into direct action rules (Anderson *et al*., 2019).

Specifically, production compilation involves the integration of two productions into a novel one, thus bypassing the retrieval of operators from declarative memory. Each newly learned production is initially assigned a utility of zero and starts competing with its original parent production. Every time the new production is selected, its utility gets progressively updated until it reaches its true value. Utility values are incrementally adjusted based on the difference learning equation, which is shown on equation 1 (see Anderson (2007) and Anderson *et al*. (2019) for further details):

$$U_i(n) = U_i(n-1) + \alpha[R_i(n) - U_i(n-1)] \quad (1)$$

where $U_i(n)$ corresponds to the nth update of the ith production utility, $R_i(n)$ corresponds to the reward at the nth update, $U_i(n-1)$ corresponds to the ith production utility at the n-1th update, and α is the learning rate. Note that α was set to 0.2 in our model.

Second, monitoring time was a crucial aspect of skill acquisition. The model kept track of time in its temporal module through a pacemaker-accumulator internal clock timing system that generated regular pulses (Taatgen, van Rijn & Anderson, 2007). Pulses were monitored in the form of time ticks whose value could be accessed via a temporal buffer (Taatgen *et al*., 2007). In the ACT-R model, time ticks were reset at the start of the game, and every time the model fired a missile.

Finally, the controller module monitored and refined the estimation of a number of key game parameters including the ship's aim and the shot timing threshold (Anderson *et al*., 2019). The controller module progressively adjusted these parameters via a control tuning mechanism starting with greater tolerance and narrowing the tolerance over time, so that values could be progressively tuned (Anderson *et al*. 2019; Seow, Betts & Anderson, 2019). One key parameter that impacted the speed of control tuning was the temperature (see equation 2).

$$T(t) = A/(1 + B.t) \quad (2)$$

where A is the initial temperature which was set to 1.0, B is a scaling factor which was set to its default value of 1/180, and t is the time in seconds that elapsed since the start of the game. In our model, control tuning critically involved two control values: First, the model needs to turn the ship so it will be aimed at the fortress when it later fires. Since the ship is moving, it searches for an offset in its aim from -18 degrees (lower bound) to 0 degrees (upper bound). Second, the ship needs to pace its fires to avoid both resets and deflations. The controller considers a range from 8 time ticks or 126 ms (lower bound) to 28 time ticks or 1476 ms (upper bound).

The range for shot timing in this model was much larger than in Anderson *et al*. (2019) because there was no information about what the appropriate time was whereas in the original *Space Fortress*, subjects were told that the lower bound is 250 ms. In *Auto Orbit*, the model narrowed the firing range to search as it gained information. Specifically, the "detect-reset" and "detect-deflate" productions were responsible for adjusting the firing time threshold range of parameters. While the time threshold's *upper* bound progressively *decreased* during deflations, the time threshold's *lower* bound progressively *increased* during resets.

Methods

Experimental Design

In this experiment, all human subjects and ACT-R models played a total of 15 games that were 3 minutes in duration (45 minutes in total). Each subject was randomly assigned to one of three possible conditions corresponding to the *fast*,

medium and *slow* game speed (see Table 1). In the fast game speed condition, the ship's orbital speed was 1.0 pixel per game tick (16 ms), and the missile speed was 10 pixels per game tick. Agents assigned to that condition needed to fire within the [250 ms – 600 ms] interval. Each fired missile that resulted into a miss was penalized by a loss of 2 points, and each balloon burst was rewarded by a gain of 100 points.

In the medium game speed condition, all aspects of the game happened at half the speed of the fast condition including the timing of shots. This was halved again for the slow game speed condition. Individuals earned points as a way to get monetarily compensated for their gameplay. The total number of points per game was our main measure of skill level which was assessed independently in each game speed. We designed a point system that met the two following criteria: First, participants should earn the greatest amount of points per balloon burst in the slowest conditions so that all players get fairly compensated for the same game length (3 minutes); Second, the point system should be adjusted such that participants in easy conditions (e.g., slow speed) may not earn significantly more than participants in hard conditions (e.g., fast speed). Each balloon burst thus led to a reward that increased by 100 points each time the game parameters were halved (see Table 1). Note that we did not compare performance scores across speeds.

Table 1: Description of the three game speed conditions

Game Speed	Speed Multiplier	Resets	Deflations	Points
Fast	1.0	250 ms	600 ms	100/burst
Medium	0.5	500 ms	1,200 ms	200/burst
Slow	0.25	1,000 ms	2,400 ms	300/burst

Agents

Human Participants We are reporting data from 60 human participants randomly assigned to each of the 3 game speed conditions. Participants were aged 21 to 40 years-old (M = 30.5, SD = 4.7). Forty were male and 20 were female. All participants were recruited on the Amazon Mechanical Turk (mTurk) online platform. Subjects earned a base pay of $4 for completing the experiment, in addition to a bonus which was proportional to their performance (in points) as specified on Table 1. On average, participants earned a bonus of $5.

ACT-R Models Ninety ACT-R model simulations were conducted in each of the 3 game speed conditions (270 model runs in total). All models were initialized with the same parameters.

Procedure

The mTurk experiment consisted of four main steps. First, participants filled out a short background questionnaire including questions about the participants' demographics. Second, they read a short description of the *Auto Orbit* video game including game play instructions. Third, participants were randomly assigned to one out of the three experimental conditions (see Table 1) and completed 15 games that lasted 3 minutes each. Finally, they filled out some additional questionnaires where they provided feedback and wrote about strategies that they used during the experiment.

Experimental Measures

In this study, we were interested in a number of experimental measures pertaining to motor skill acquisition. Our main dependent variable was *performance,* which was operationalized as points earned per game (see Table 1). The design comprised a total of 4 independent variables: the keypress sequence *entropy*, the *inter-shot-interval (ISI) coefficient of variation logarithm,* and the shot *periodicity* and *regularity*. All measures were computed across game cycles ("balloon respawn" to "balloon burst") without random rotations for every agent and every game.

Entropy The entropy was our measure of keypress sequence regularity in *Auto Orbit*. We focused on the relative frequency of various keypress triples. With three keys ('F': fire, 'L': left, 'R': right) there are $3^3 = 27$ triples. We computed the proportions of each keypress triple per game by using a non-overlapping counting method (Python count() function) with Laplace smoothing for each keypress triple in all game cycles. We used the Shannon entropy measure, which quantifies unpredictability of information content in a probability distribution (Shannon & Weaver, 1949). Shannon entropy's formula is given in equation 3:

$$H(X) = -\sum_{i=1}^{27} p_i \,.\, log_2\, p_i \qquad (3)$$

where X refers to a game number and p_i refers to the probability of the ith triple. This entropy measure could vary from 0 (only 1 triple throughout) to 4.75 (all triples equally likely). We expected the entropy measure to decrease as subjects developed a systematic approach to the game.

Log CV Inter-Shot-Interval (ISI) In order to measure shot timing variability, we extracted the time interval between consecutive shots in milliseconds, named inter-shot-interval (ISI) within game cycles. For each game of every agent, we then made use of the coefficient of variation (CV), which is defined as the standard deviation divided by the mean of the ISIs, consistent with previous research (Loehr & Palmer, 2009). An average CV of the ISIs was computed across game cycles within each agent's game. Because our measure of CV and our performance variable were not linearly related, we carried out data transformation on CV and calculated its logarithm instead. We expected this measure to decrease as subjects became more skilled.

Periodicity and Regularity The shot regularity measure was computed based on the shots autocorrelation function within games. This method has been used in the music

information retrieval literature to perform meter extraction (Brown, 1993). For each game cycle, we first re-preprocessed agents' log files such that we would get a single discrete time series of fire events, where individual entries corresponded to successive game ticks of 16 ms. At every tick, a 1 indicated a fire keypress hold event and a 0 indicated a fire keypress release event. We could then compute the correlation coefficient (Box & Jenkins, 1976) of keypress actions at a particular lag (see equation 4) using the 'acf' function from the statsmodels time series analysis ('tsa') library in Python (McKinney, Perktold & Seabold, 2011):

$$r_l = \frac{\sum_{i=1}^{100-l}(x_i - \bar{x})(x_{i+l} - \bar{x})}{\sum_{i=1}^{100}(x_i - \bar{x})^2} \quad (4)$$

where l refers to the current time lag. A total of 100 time lags of 16 ms were investigated in each autocorrelation function. We averaged the autocorrelation function across game cycles of each agent's game. As a result, for each agent, we obtained 15 autocorrelation functions corresponding to each of the 15 games. Figure 2 displays an example of a game autocorrelation function in a subject. Positive peaks in this function reflect lags at which the fire keys tended to be pressed.

Finally, we used each game autocorrelation function to extract our two measures of interest: *periodicity* and *regularity*. To do so, we identified the first non-zero lag positive peak of the autocorrelation function. Periodicity was the lag for this peak at which fires tended to be pressed. Regularity was the height of this function and reflected how consistently keypresses occurred at this lag. In our example, the first non-zero autocorrelation positive peak has been identified with a red bar (see Figure 2). We expected that better play would be associated with decreased periodicity as subjects got their shots closer to the threshold, and increased regularity as they got more consistent in their timing.

Data Analysis

Our data set consisted of the above measures, each recorded once per game and per agent (i.e., in humans and models).

Linear Mixed-Effects Model In order to assess each measure's individual effect on skill acquisition, we fit a linear mixed-effects model (LMEM) across game speeds. Our main dependent variable was performance, operationalized as points earned per game. Our predictors were the four measures described earlier, each modeled as a fixed factor. In addition to our fixed factors, we added two random factors to account for some of the variability in our performance measure that was not explained by our four linear predictors. The first random factor accounted for differences across participants' skill levels and was modeled as a random intercept. The second random factor accounted for residual variance in performance related to individual game numbers that could not be captured by our four fixed factors.

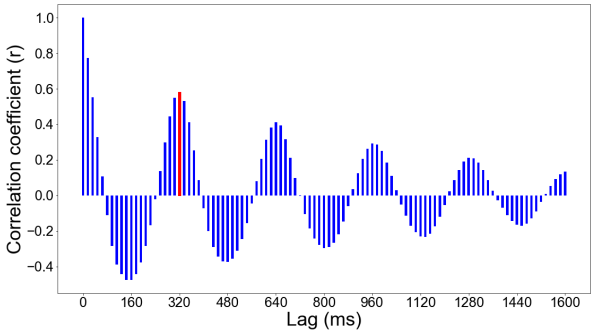


Figure 2: Autocorrelation function in game 11 of one of the subjects in the fast speed condition. The red bar indicates the periodicity (lag) and regularity (correlation coefficient) of this subject's shot timings in that game.

In R, we used the 'lme4' (Bates *et al.*, 2014) package to fit our linear mixed-effects model. The model was written as *lmer*(*Performance* ~ *Entropy* + *logCV* + *Periodicity* + *Regularity* + (1|*Subject*) + (1|*GameNb*)). For each model, the 95 % confidence interval was computed for each estimate using bootstrapping with resampling ('bootMer' function in R). A total of 1000 simulations were run for each bootstrapped 95 % confidence interval.

Results

Behavioral Results

We hereby present the results from human games and ACT-R model simulations across the three game speed conditions (fast, medium, and slow). For each measure of interest, we report the mean within games across agents along with the standard error. Because the ACT-R models were all initialized with the same parameter values, there is lower variability among models than humans for each measure.

We first report the performance results across games in humans and models (see Figure 3). Humans and models achieved similar numbers of points, both showing rapid initial improvement approaching an asymptote by 15 games. However, some differences are worth noting: In the fast speed condition, humans performed somewhat better than the models. In the medium and slow speed conditions, models had a somewhat steeper slope than humans.

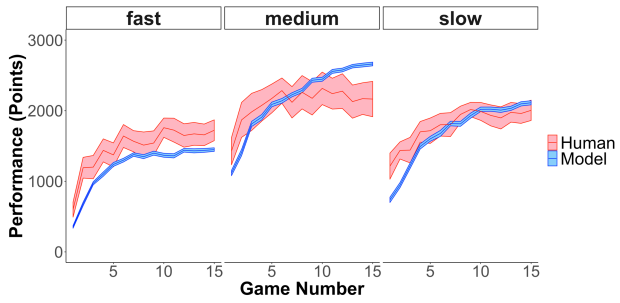


Figure 3: Performance scores over the games. Human performance is indicated in red (N = 20 per speed); model performance is indicated in blue (N = 90 per speed). Shaded areas indicate the standard error of the mean (S.E.M).

In terms of keypress sequencing, both models and humans showed similar levels of entropy with an increase for slower games. (Figure 4a), but humans' entropy progressively decreased in early games towards their asymptote whereas models' entropy was constant across all games. We think this reflects the fact that models have a constant strategy, whereas subjects' strategies are evolving during the early games and they only settle down to a constant strategy in later games.

The results with respect to ISI variability (Figure 4b), are similar to entropy. While models show some decrease over games, subjects show a more drastic decrease with a large early drop in the 2 first games. We think this reflects subjects' evolving strategy and progressive adaptation to the game's shot timing constraints. As such, early changes will produce large changes in motor timing.

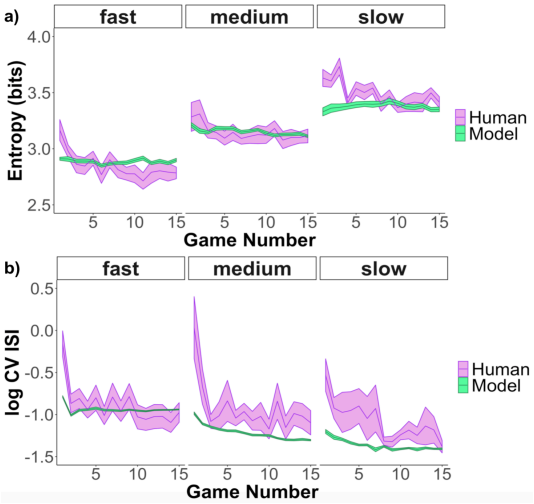


Figure 4: a) Entropy over the games in humans and models. b) Logarithm of the inter-shot intervals (ISI in ms) coefficient of variation (CV). Shaded areas indicate the standard error of the mean (S.E.M).

Analysis of shot timing autocorrelations provided periodicity and regularity measures (Figure 5). With regards to the shot periodicity, humans and models were quite similar. Both had shot timings within the assigned game speeds' firing intervals (see Figure 5a), and both models and subjects showed an increase in periodicity in the slow speed condition. (see Figure 5b). That increase reflected an early tendency to fire too soon, which both models and humans had to learn to change.

Both models and humans showed an increase in regularity over games, but human regularity increased as the game speed got faster whereas model regularity did not increase with speed. The regularity in the model reflects its timing mechanism, which is a variant of the attentional gate model. That model produces a variability that scales with duration, so that regularity will not change much. In contrast, subjects may be changing their timing processes as they move to timing actions well under a second.

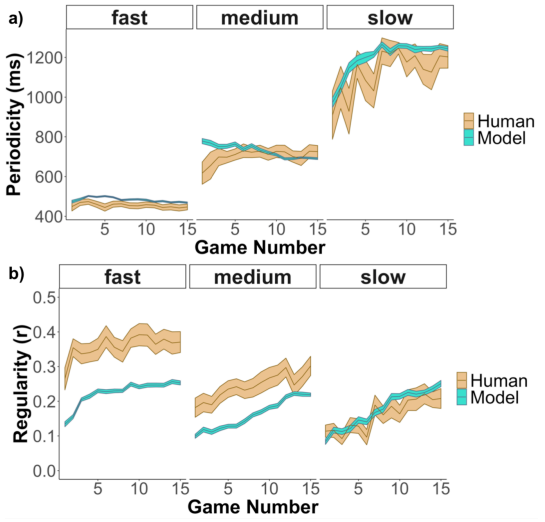


Figure 5: a) Periodicity over the games in humans and models. b) Regularity over the games. Shaded areas indicate the standard error of the mean (S.E.M).

Linear Mixed-Effects Model

We hereby report the LMEM results pertaining to humans and ACT-R models separately. Table 2 displays the human LMEM results. Three models were fitted to each game speed data set. Entropy and the logarithm of the ISI coefficient of variation were both predictive of performance across all speeds. A decrease in entropy was reliably predictive of performance, and both lower and upper bounds of the confidence interval were negative. Similarly, a decrease in ISI variability was also reliably predictive of performance across all speeds. With respect to the shots autocorrelation measures, a positive effect of regularity (more regular) on performance and negative effect of periodicity (faster firing rate) on performance were found in the fast and medium game speeds, but *not* in the slow game speed. All significant effects are in line with our expectations about these factors. The fact that regularity and periodicity are not predictive in the slow game speed reflects the fact that it is not time pressured.

Table 2: Human LMEM results across game speeds

	Game Speed - Human					
	Fast		*Medium*		*Slow*	
	Estimate	95 % CI	Estimate	95 % CI	Estimate	95 % CI
Entropy	-490^{***}	$(-638, -312)$	-641^{***}	$(-844, -421)$	-295^{***}	$(-452, -136)$
Log CV ISI	-164^{***}	$(-242, -84)$	-84^{*}	$(-161, -12)$	-146^{***}	$(-206, -83)$
Regularity	1171^{***}	$(732, 1650)$	1062^{**}	$(469, 1677)$	378	$(-141, 943)$
Periodicity	-1.31^{***}	$(-2.25, -0.38)$	-0.52^{***}	$(-0.87, -0.15)$	0.10	$(-0.03, 0.23)$
Adjusted R^2	0.88		0.88		0.79	

$^{***}p < .001$; $^{**}p < .01$; $^{*}p < .05$

Table 3 displays the ACT-R models' LMEMs. We found a consistent positive effect of shot regularity on performance across all three game speeds. In contrast to

Table 2, there are a number of significant effects in the opposite direction of expectation. Increased periodicity is associated with more points in the slow condition. We think this is associated with the initial too-fast firing in the slow condition (Figure 5) and is partly responsible for the lack of an effect for humans in Table 2. The other discrepancy is that greater ISI variability was predictive of higher performance in the fast and slow game speeds. This is puzzling, because in terms of simple correlation, there is no correlation between points and this measure in the fast game (r = .005) and a weak negative correlation in the slow games (r = -.196). The direct correlation for medium speed, where Table 3 shows the expected negative effect, is .581.

Table 3: ACT-R models LMEM results across game speeds

	Game Speed - ACT-R models					
	Fast		*Medium*		*Slow*	
	Estimate	95 % CI	Estimate	95 % CI	Estimate	95 % CI
Entropy	-451^{***}	$(-531, -372)$	-15	$(-118, 97)$	53	$(-22, 127)$
Log CV ISI	1034^{***}	(896, 1177)	-372^{***}	$(-531, -205)$	604^{***}	(477, 734)
Regularity	974^{***}	(745, 1201)	1043^{***}	(754, 1348)	558^{***}	(326, 822)
Periodicity	-0.05	$(-0.40, 0.27)$	-0.62^{***}	$(-0.82, -0.43)$	0.32^{***}	(0.21, 0.44)
Adjusted R^2	0.73		0.66		0.65	

$^{***}p < .001$; $^{**}p < .01$; $^{*}p < .05$

Discussion and Conclusion

ACT-R models and humans showed similar improvement scores in the *Auto Orbit* video game. Motor skill acquisition was characterized by a fast performance increase in early games, and a slower learning rate in later games where performance progressively plateaued towards an asymptote. In terms of motor behavior, humans and models both learned to be more regular and less variable in terms of keypress sequential patterns and shot timing. One challenge in the *Auto Orbit* video game was for players to learn how to shoot within a firing interval bounded by resets and deflations. Shot timing autocorrelation analyses revealed that humans and models learned to fire missiles within their assigned game speed firing interval (periodicity) with increasing rhythmicity (regularity).

Nevertheless, our analyses also revealed a number of motor learning differences between humans and models which are worth discussing. First, we found that humans' keypress patterns and shot timing were more variable than models, particularly in early games, but quickly converged towards models' variability levels as performance increased. These variability patterns in humans fit with previous neuroscience (Wu *et al.,* 2014) and motor skill learning research (Caramiaux *et al*., 2018) suggesting that motor and timing variability may predict performance over the course of motor skill acquisition.

Second, while shot regularity increased in a similar fashion at the slow speed across humans and models, we found that human subjects' shot regularity levels were higher at faster speeds, whereas models' regularity levels remained constant across speeds. This result may be due to a higher reliance on motor circuits (Lewis & Miall, 2003) and subcortical modulation (Ivry & Spencer, 2004; Koch *et al*., 2007) under fast speed constraints. Specifically, past research suggests that motor timing tasks involving fast discrete actions such as repetitive keypresses may heavily recruit the cerebellum through dynamic sensorimotor learning and motor error correction (Koch *et al*., 2007; Breska & Ivry, 2016).

While model performance and motor learning were relatively close to humans, improvements can be made to optimize the ACT-R model and better simulate human motor skill acquisition. One option would be to change the parameterization of ACT-R to make the model more variable, both within- and between-models. In terms of within-models variability, one could vary the noise levels and learning rates related to different components of the model. One example is the temporal module noise parameter whose increase may lead to further shot timing variability in ACT-R. Another example is the utility learning and production compilation learning rate (α in equation 1), which controls the speed at which newly formed productions replace their original parent productions. Specifically, high values of α typically lead to a faster rate of production compilation and skill acquisition whereas low values of α typically lead to a lower rate of production compilation and skill acquisition. One last example is the initial temperature (A in equation 2), whose value assignment may lead to different degrees of randomness in control tuning. Generally, lower initial temperatures enable the model to incorporate more of its learning experience into its game play, but they also increase the risk of converging towards non-optimal values if the initial temperature is too low.

As to between-models variability, past research by Anderson *et al*. (2019) explored potential performance fluctuations related to the adjustment of a few selected parameters. Specifically, the authors explored the effects of α (see equation 1) in the [0.025, 0.3] range, and the initial temperature (A in equation 2) in the [0.1, 2.0] range. It would be of interest to further explore the stochastic initialization of these parameters in our current ACT-R model to determine whether one could replicate humans' inter-subject variability patterns.

A second option would be to modify the initialization of operators to enable the model to adjust its behavior to a fast vs. slow game speed depending on shot timing threshold information in the controller module. Alternatively, one could vary the order in which operators are retrieved at different phases of the game such that the model would initially prioritize shooting over aiming, but would progressively switch to a more optimal strategy that prioritizes aiming over shooting.

Last but not least, our current results strongly suggest that ACT-R's current motor module needs the ability to adjust its motor behavior according to the speed at which it executes motor actions. One striking result was that human shot timing became increasingly rhythmic as the game

speed got faster. One way of modeling this time-related effect of speed on motor behavior would be to augment the current motor module with its own timing component, such that the model would fire missiles with increasing rhythmicity at faster speeds. This novel addition would fit with ACT-R being a template of human behavior.

In sum, we have shown that human motor skill learning was characterized by time-independent and time-dependent effects of speed. On the one hand, variability in keypress sequencing and motor timing were shown to predict skill acquisition regardless of the game speed. On the other hand, motor timing regularity and periodicity were shown to only be predictive of performance in the sub-second range. As a way to model these effects in ACT-R, we suggest a number of improvements which include incorporating a timing component into ACT-R's motor module.

Acknowledgements

This research was supported by the Office of Naval Research Grant N00014-15-1-2151 and AFOSR/AFRL award FA9550-18-1-0251. We would like to thank Dan Bothell for his help with the implementation of the *Auto Orbit* video game and the ACT-R model.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?*. New York, NY: Oxford University Press.

Anderson, J. R., Betts, S., Bothell, D., Hope, R., & Lebiere, C. (2019). Learning rapid and precise skills. *Psychological Review, 126*, 727–760.

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2014). Fitting linear mixed-effects models using lme4. *J Stat Softw* arXiv:1406.5823.

Beukema, P., Diedrichsen, J., & Verstynen, T. D. (2019). Binding during sequence learning does not alter cortical representations of individual actions. *Journal of Neuroscience, 39*, 6968-6977.

Box, G. E. P., & Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. San Francisco, CA: Holden-Day.

Breska, A., & Ivry, R. B. (2016). Taxonomies of timing: where does the cerebellum fit in?. *Current opinion in behavioral sciences, 8*, 282-288.

Brown, J. C. (1993). Determination of the meter of musical scores by autocorrelation. *The Journal of the Acoustical Society of America, 94*, 1953-1957.

Caramiaux, B., Bevilacqua, F., Wanderley, M. M., & Palmer, C. (2018). Dissociable effects of practice variability on learning motor and timing skills. *PloS one, 13*: e0193580.

Coull, J. T., Cheng, R. K., & Meck, W. H. (2011). Neuroanatomical and neurochemical substrates of timing. *Neuropsychopharmacology, 36*, 3-25.

Decety, J., Jeannerod, M., & Prablanc, C. (1989). The timing of mentally represented actions. *Behavioural brain research, 34*, 35-42.

Diedrichsen, J., & Kornysheva, K. (2015). Motor skill learning between selection and execution. *Trends in cognitive sciences, 19*, 227-233.

Ivry, R. B., & Spencer, R. M. (2004). The neural representation of time. *Current opinion in neurobiology, 14*, 225-232.

Koch, G., Oliveri, M., Torriero, S., Salerno, S., Gerfo, E. L., & Caltagirone, C. (2007). Repetitive TMS of cerebellum interferes with millisecond time processing. *Experimental brain research, 179*, 291-299.

Lewis, P. A., & Miall, R. C. (2003). Distinct systems for automatic and cognitively controlled time measurement: evidence from neuroimaging. *Current opinion in neurobiology, 13*, 250-255.

Loehr, J. D., & Palmer, C. (2009). Sequential and biomechanical factors constrain timing and motion in tapping. *Journal of Motor Behavior, 41*, 128-136.

Mané, A., & Donchin, E. (1989). The space fortress game. *Acta psychologica, 71*, 17-22.

McKinney, W., Perktold, J., & Seabold, S. (2011). Time series analysis in Python with statsmodels. *Jarrodmillman. Com*, 96-102.

Palmer, C., & Pfordresher, P. Q. (2003). Incremental planning in sequence production. *Psychological Review, 110*, 683-712.

Seow, R. Y. T., Betts, S., & Anderson, J. R. (2019). Transfer effects of varied practice and adaptation to changes in complex skill acquisition. In Stewart, T.C. (Ed.). *Proceedings of the 17th International Conference on Cognitive Modelling* (pp. 222-227). Waterloo, Canada: University of Waterloo.

Shannon, C. E., & Weaver, W. (1949). *The mathematical theory of communication*. Urbana, IL: University of Illinois Press.

Taatgen, N. A., Van Rijn, H., & Anderson, J. (2007). An integrated theory of prospective time interval estimation: The role of cognition, attention, and learning. *Psychological Review, 114*, 577-598.

Verstynen, T., Phillips, J., Braun, E., Workman, B., Schunn, C., & Schneider, W. (2012). Dynamic sensorimotor planning during long-term sequence learning: the role of variability, response chunking and planning errors. *PLoS One, 7*: e47336.

Wu, H. G., Miyamoto, Y. R., Castro, L. N. G., Ölveczky, B. P., & Smith, M. A. (2014). Temporal structure of motor variability is dynamically regulated and predicts motor learning ability. *Nature neuroscience, 17*, 312-321.

Zakay, D., & Block, R. A. (1996). The role of attention in time estimation processes. *Advances in Psychology*, *115,* 143–164.

Does ACT-R Model Me?

Emily Greve (emilygreve@cmail.carleton.ca)
Institute of Cognitive Science, 1125 Colonel By Dr.
Ottawa, ON. K1S 5B6 CA.

Elisabeth Reid (elisabethwood@cmail.carleton.ca)
Institute of Cognitive Science, 1125 Colonel By Dr.
Ottawa, ON. K1S 5B6 CA.

Robert West (robert.west@carleton.ca)
Institute of Cognitive Science, 1125 Colonel By Dr.
Ottawa, ON. K1S 5B6 CA.

Abstract

We were interested in testing Newell's Micro Strategies hypothesis as well as assumptions made by both ACT-R and SGOMS theory using a mobile game and a predictive SGOMS-ACT-R model. The Model is designed to predict expert game play. We found in most conditions the model did predict the results, however in one condition the player employed an alternative Micro Strategy.

Keywords: SGOMS; GOMS; ACT-R; Micro-cognition

Introduction

Often in studies on cognitive tasks many participants are used, and their results are averaged together to deal with variation in the results. This variation is generally interpreted as noise. However, Newell (1973) said that by averaging over many participants, we may be averaging over different strategies. If you control noise by training participants to the expert level, so they are on the same place on the learning curve, and rigorously control the task, different patterns in the data can be attributed to different Micro Strategies. Micro Strategies take place at the millisecond scale and can vary during simple cognitive tasks (Gray and Boehm-Davis, 2000).

In *You Can't Play 20 Questions with Nature and Win*, Newell (1973) notes the importance of understanding the different micro strategies for using our cognitive, perceptual, and motor systems to perform tasks in Cognitive Psychology experiments (note, Newell refers to these strategies as Methods, but we will use the term, Micro Strategies, and reserve the term, Methods, for use in our SGOMS model). For example, if a person needs to respond to a stimulus by typing a two-letter code, they could use one finger, one finger on each hand, or two fingers on the same hand if the letters were close together. If they know what code to type, they only need to see that a stimulus appears; they do not have to wait to fully recognize it before responding. However, some people might wait to fully register the identity of the stimulus before responding. Newell's point was that we should not average across different strategies as it produces meaningless numbers that do not accurately reflect the operation of the underlying cognitive system.

A previous study by West, Ward, Dudzik, Nagy, & Karimi (2018) used an ACT-R Agent (Anderson & Lebiere, 1998) built according to SGOMS (West & Nagy, 2007). The Agent they built was designed as a predictive model on expert gameplay. Only two participants were used but they were extensively trained to be high performing experts. The results of the study showed that two participants matched each other and the ACT-R Agent within milliseconds of accuracy under specific conditions, but not under others.

To further the research started by West et al. (2018), we decided to follow a similar experimental design but developed a version of the game without the conditions where the two subjects varied.

The Game

The new game, called Four Button was built using MIT app inventor 2 (https://appinventor.mit.edu/), and is run as an app on mobile devices. Four Button Expert levels follow the SGOMS structure and explicitly uses a hierarchy structure composed of operators (individual button presses), methods (fixed series of button presses), Unit Tasks, and Planning Units. Four Button Expert can be best explained by comparing gameplay to that of a First-Person Shooter videogame, such as. Dead Space, a videogame in which a player must fight through different levels of a game. Throughout the levels players encounter aliens which they must fight using different combinations of moves and weapons.

The Methods Level

The Methods level is equivalent to knowing which buttons correspond to which actions of the character. Buttons on a game controller such as X, O, and R2 correspond to actions such as Jump, duck, and shoot. In the Four Button Expert the Methods take the form of a two-letter prompt and a corresponding four-digit response. Players must enter the four numbers when prompted by the appearance of the two letters at the top of the screen (see Table 1). Expert players would have the four-number sequence proceduralized and be able to enter it immediately when they know which Method is required.

Table 1: The Methods

Methods	
AK	1234
SU	4123
ZB	2143
RP	4321
FJ	3214
HW	2341
YP	3412
WM	1432

The Unit Task Level

The Unit Task level is equivalent to knowing the different action sequences you must use to fight an enemy. For example if there is one alien type that you shoot until he executes/performs an attack on you upon which your you duck before resuming shooting, your button sequence would be R2, R2, O, R2. Compare this to another alien type where you must shoot, jump to avoid his attack type, shoot and then duck, in this case your button sequence would be R2, X, R2, O. Players must use a different sequence of the same actions in the different conditions. In our game the two-letter prompts are organized in specific and consistent sequences which then correspond to unit tasks (see Table 2). Expert players recognize that specific Methods signify the beginning of a Unit Task and know which related sequence of numbers are needed to complete the Unit Task. In two of the Unit Tasks (RP and HW) there are splits that occur, where one of two or one of three Methods could be displayed. The splits always occur at the same place in each Unit Task. This is the equivalent of some of the aliens having two or three different possible attack types that they employ at random at a certain point in the sequence.

Table 2. The Unit Tasks

Unit Tasks
Unit Task 1 (AK UT)
AK-WM-SU-ZB-FJ
Unit Task 2 (RP UT)
RP-SU< ZB-WM / YP-FJ
Unit Task 3 (HW UT)
HW-YP< FJ / ZB / SU

The Planning Unit Level

The Planning Unit level can be compared to a full level of our hypothetical video game. During a level, different aliens would be activated at different points throughout the level. An expert at this game would know that on level 1, Alien type 1 appears, followed by Alien type 2 and followed by Alien Type 3. Whereas on Level 2 the Alien order is Type 3, 1, then 2. Expert Players would be able to know exactly which order the Unit Tasks (Aliens) appear and which sequence of Methods (actions) take place within those. Our game follows this structure as well, where each Planning Unit holds the same Unit Tasks in different orders (see Table 3). Expert players are able to recognize which Planning Unit they are in by looking at the first Method code of the planning unit.

Table 3. The Planning Units

The Planning Units
Planning Unit 1
(AK UT)-(HW UT)-(RP UT)
Planning Unit 2
(RP UT)-(HW UT)-(AK UT)
Planning Unit 3
(HW UT)-(RP UT)-(AK UT)

Methodology

For our study we had 1 participant in order to thoroughly understand one individual's micro strategies before collecting more. Also, we could compare the results to similar conditions in the previous version of the game (West et al., 2018). This Methodology of making detailed comparisons between a few participants has also been successfully used by Gray and Boehm-Davis (2000), and Shiffrin and Cousineau (2004).

The game app was downloaded onto the participant's phone. The participant learned the game, starting at the Methods level. They moved up to the Unit Task level after they could confidently play each Method and their timing was consistent across all 8 Methods. Once they could play all three Unit Tasks they were moved up to the final level, the full game or Planning Unit level. Reaction time was based on how fast it takes for the participant to enter the corresponding four-digit code from when the two-letter code first appears on the screen.

Predictions

Based on the ACT-R model and previous results from West et al. (2018) we were able to make some predictions. For the conditions with multiple possible responses, we predicted that Hick's law, which states that reaction times increase as the number of stimulus–response alternatives increase (Hick 1952), would not apply. That is, we predicted no difference between our 3 choice and 2 choice conditions (see table 2). This is because the model assumes expert players will not rely on declarative memory (see Schneider and Anderson, 2011).

The model has one free parameter, which is the perceptual motor time to respond. However, this parameter is different depending on whether the next method could be memorized (known), or whether it was necessary to see the code before choosing the method (unknown). Because the known and unknown conditions have different estimated parameter values, we treated them separately. For the unknown condition the model predicts that a method signaling the

beginning of a planning unit will take more time due to the SGOMS overhead required to keep track of the planning unit, compared to a method not associated with the beginning of a planning unit,. For the known condition, the model predicts that a method signaling the beginning of a unit task will take more time due to the SGOMS overhead required to keep track of the unit task, as opposed to a method not associated with the beginning of a unit task.

Finally, we predicted that our results should be the same as West et al. (2018) when scaled so that the known and unknown parameter values are the same across all subjects and the model.

Results

Methods where the player's data had errors were removed because we are interested in the player's time during conditions when they are playing correctly. To clean the data, it was sorted from smallest to largest time (in milliseconds). Outliers were cut off by detecting a knee in the data. (Satopaa, Albrecht, Irwin, & Raghavan, 2011). The mean average time was calculated from the remaining data in each condition.

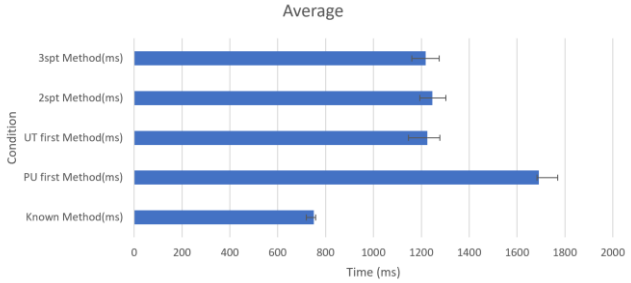


Figure 1: The Player Results (ms) under the different conditions with confidence intervals (0.05)

Figure 1 shows the player results with 0.05 confidence intervals. As predicted the three split (3spt method) and two split (2spt method) conditions were the same. Additionally, the predictions of extra processing time for unknown methods at the start of a planning unit (PU first method) and known methods at the start of a unit task (UT first method) was supported. This is illustrated by comparing PU first method to both 3spt method and 2spt method and comparing UT first method to known method.

Compared to the West et al. (2018) results, these results were scaled by assuming that differences in speed were due to perceptual/motor differences only. The model treats perceptual/motor as an additive factor, so we used the split conditions (which were combined into one condition) and the known methods condition to estimate the difference between participants in perceptual/motor speed for known perceptual motor actions and unknown perceptual motor actions. We then equalized perceptual/motor speed by subtracting an amount so that all participants were the same as the fastest participant in these two conditions, whose perceptual motor times were also used in the model. To test the model this same amount was also subtracted from the planning unit start condition (Pu first method) and the unit task start condition (UT first method), with the prediction that model and participants also be the same across these conditions.

The modeling results, displayed in Figure 2, show that our participant matched the model and the West et al. (2018) participants for the planning unit start condition, but took significantly more time for the unit task start condition. However, the time for the unit task start condition closely matched the unknown method time, suggesting that our participant cued off the displayed code rather than using their memory for this condition.

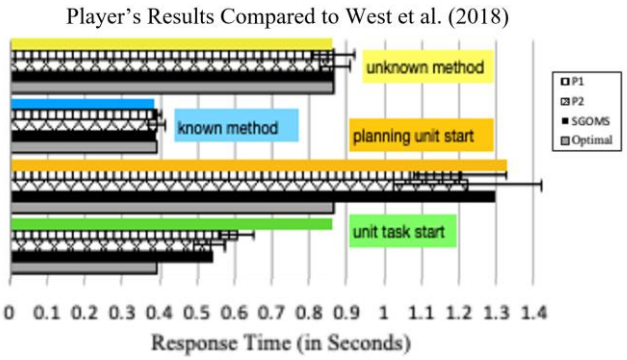


Figure 2. The data from our Participant compared to the result of the previous study. The hatched bars are the other two players from West et al. (2018), the black bar is the SGOMS/ACT-R model predictions and the gray bar is the optimal ACT-R prediction, where the model does not keep track of where it is in the task

Conclusion

Overall, we showed support for the idea that data and models can be used to study micro strategies in individuals. In particular, as in Gray and Boehm-Davis (2000) and Shiffrin and Cousineau (2004), we provide evidence that people can adopt different micro strategies even for simple tasks.

We can also make a prediction: If our analysis is correct, it should be possible to alter our participant's strategy by training them to rely on memory rather than vision for the unit task start condition. Such training should produce the predicted result. We will attempt this training and report the results at a future conference.

References

Anderson, J. R., & Lebiere, C. (1998). The atomic components of thought. Mahwah, NJ: Erlbaum.

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of experimental psychology: Applied*, *6*(4), 322.

Hick WE. (1952) On the rate of gain of information. Quarterly Journal of Experimental Psychology. 4:11–26.

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium.

Satopaa, V., Albrecht, J., Irwin, D., & Raghavan, B. (2011, June). Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops* (pp. 166-171). IEEE.

Schneider, D. W., & Anderson, J. R. (2011). A memory-based model of Hick's law. *Cognitive psychology*, *62*(3), 193–222.

Shiffrin, R., & Cousineau, D. (2004). Termination of a visual search with large display size effects. *Spatial vision*, *17*(4), 327-352.

West, R. L., & Nagy, G. (2007). Using GOMS for modeling routine tasks within complex sociotechnical systems: Connecting macrocognitive models to microcognition. *Journal of Cognitive Engineering and Decision Making*, *1*(2), 186-211.

West, R., Ward, L., Dudzik, K., Nagy, N., & Karimi, F. (2018, July). Micro and Macro Predictions: Using SGOMS to Predict Phone App Game Playing and Emergency Operations Centre Responses. In *International Conference on Engineering Psychology and Cognitive Ergonomics* (pp. 501-519). Springer, Cham.

One Size Doesn't Fit All: Idiographic Computational Models Reveal Individual Differences in Learning and Meta-Learning Strategies

Theodros Haile (theodros@uw.edu)
Department of Psychology, University of Washington
Campus Box 3515525, Seattle, WA 98195 USA

Chantel S. Prat (csprat@uw.edu)
Department of Psychology, University of Washington
Campus Box 3515525, Seattle, WA 98195 USA

Andrea Stocco (stocco@uw.edu)
Department of Psychology, University of Washington
Campus Box 3515525, Seattle, WA 98195 USA

Abstract

In humans, learning is a complex phenomenon that depends on the joint contribution of multiple interacting systems, most notably memory (WM), long-term memory (LTM) and reinforcement learning (RL). There are vast individual differences in learning mechanism deployment. It is also, often, difficult to assess, through behavioral measures, the relative contributions of these systems during learning as well the specific strategies individuals rely on in performing a task. Collins (2018) put forward a working memory-reinforcement learning combined model that addresses these issues within a simple domain, but largely ignores the long-term memory component. In this project, we built four (two single-mechanism RL and LTM, and two integrated RL-LTM) idiographic learning models based on the ACT-R cognitive architecture. We aimed to examine individual differences and estimate parameters that could explain preferential use of learning mechanisms using the Collins (2018) stimulus-response association task. We found that different models provided best-fits for individual learners with more variability in learning and memory parameters observed even within the best fitting models. Our conclusion is that irreducible differences in learning and meta-learning strategies exist within individuals even within relatively simple tasks, and that model-based approaches are necessary to characterize and explain behavioral data.

Keywords: Individual differences; reinforcement learning; ACT-R; working memory; declarative memory; learning.

Introduction

Individual differences in the ability to learn new associations are foundational to most measures of aptitude— a construct that describes the readiness with which one can acquire a complex skill. But even basic associative learning paradigms, like stimulus-response mappings, have been shown to rely on a mixture of learning mechanisms including working memory, reinforcement learning, and long term memory (Stocco et al., 2010). Though a considerable amount of research has investigated how task characteristics drive these mechanisms during learning (Collins & Frank, 2012), less work has been devoted to understanding how and when they may be deployed differently in different learners. To examine this, we built two single-mechanism and two multi-system stimulus-response learning models using the Adaptive Control of Thought - Rational (ACT-R) cognitive architecture, and used them to examine individual learning mechanisms for the same learning task. Specifically, Anne Collins' Reinforcement Learning Working Memory task (RLWM task: Collins, 2018) was used as the task paradigm .

It can be difficult to assess the independent contributions of these learning mechanisms behaviorally. Modelling is a robust approach to evaluating the independent contributions of these mechanisms (Collins, 2018). This method further allows us to estimate individual parameters that would give us insight into the cognitive properties that resulted in different forms of skill acquisition (Daw, 2011). We adopted the RLWM task because it provided a single experiment with simple manipulations to dissociate learning mechanisms.

But in the task's simplicity lies a difficulty: long-term memory and reinforcement learning guide actions and responses that are nearly indistinguishable in the context of the task using behavioral outcomes only. In the RLWM task, participants are asked to learn associations between images (e.g. objects, shapes, and colors) and key responses through trial-and-error with feedback. The task, as designed by Collins, sought to quantify the relative contributions of working memory and reinforcement learning through two training conditions over 14 blocks–a working memory, resource-sparing, 3-image condition for 8 blocks and a resource-intensive, 6-image condition for 6 blocks. After training, participants performed an unrelated, 10-minute distractor task followed by a surprise test block. Collins et al. expect that that the 3-image associations, learned quickly through working memory, should not be remembered after the distracting break, whereas the 6-image associations, acquired through reinforcement learning, should be retained after the break and demonstrated during the test phase. This largely aligns with what we know about the durability of reinforcement learning (Stocco et al., 2010). Collins has demonstrated that

learning object-letter associations most probably occurs through the interaction of Reinforcement learning (RL) and Working Memory (WM) using a combined, interacting (RL+WMi) model (Collins 2018; Collins & Frank, 2012). They hypothesized that the fast-learning (high learning-rate) WM resource, which is limited in capacity and decays rapidly, represented by a decay parameter, cooperatively interacts with the RL portion of the model, directly influencing the computation of the reward prediction error. This model contributes less to reward prediction error when the set size is high. This model fit participant data best compared to other, RL and non-interacting RL+WM models (Collins, 2018).

One critical limitation of Collin's original modeling effort is that it implicitly assumes that all long-term associations between stimuli and responses are stored in a procedural, RL-based system, and, conversely, that all of the explicit representations of the correct responses must fit within a temporally constrained working store. This is apparent in the assumption, for example, that performance after a 5-minute interval must reflect the RL system only (Collins, 2018). Instead, our replication of the experiment shows that participants have also used their long-term declarative memory. Upon completion of the main task, participants in our study were also asked to answer the open-ended question, "*Do you recall using a specific strategy to learn the images?"* A substantial number of them reported, for instance, relying on colors, names, or other salient features of the stimuli to remember the corresponding responses. Many answers followed the common pattern "*Pictures 'A' and 'B' shared an attribute and were both associated with the keyboard response 'V', so they were grouped together".* An informal evaluation of these responses lent a trickle of confidence to the use of a possible LTM strategy, as well as the fact that participants seem to explicitly control they learning strategies Additionally, we have observed clear individual differences in learning as well as demonstration of learned associations in our subjects that stray away from the WM-RL dichotomous view of learning. For instance, a proportion of our subjects learned quickly in both object-set conditions, suggesting working memory use, but also showed that learned associations prevailed after the 10-minute break (Figures 4 and 5).

To further complicate the story, Collins' model relies on a simplified working memory system, which, in essence, is a fixed-capacity storage with fading contents. This is exactly how short-term memory was originally conceptualized by Atkinsons and Shiffrin (1968) and, while useful as a modeling tool, it is also known to be inadequate. Critically, contemporary theories think of working memory as a process arising from the interaction between attention and the strategic retrieval of long-term memory information (Kane et al., 2001; Miller, Lundqvist, & Bastos, 2018). In essence, Collin's modeling efforts confound the temporal axis of learning (long vs. short term representations) with the learning representation (implicit and procedural, driven by RL, and explicit, driven by WM).

The ACT-R Cognitive Architecture

To capture the interplay between reinforcement learning, long-term memory, and working memory within an integrated model, we decided to follow an alternative approach and build a series of models using the ACT-R cognitive architecture (Anderson, 2007). ACT-R was an obvious choice for this study because of its expansive, flexible and manipulable integration of cognitive mechanisms. In ACT-R, knowledge is represented in two possible formats, procedural and declarative. Procedural knowledge is represented as procedural rules, is identified with the basal ganglia, and is learned through reinforcement learning (Stocco, Lebiere, Anderson, 2010; Ceballos, Stocco, Prat, 2020). Declarative knowledge is represented in explicit memories. Explicit memories decay over time, but their activation can be momentarily increased through spreading activation, an attentional mechanism that can be used to maintain information for a brief amount of time and predicts individual differences in working memory capacity (Daily et al 2001). Finally, ACT-R is a realistic "end-to-end" modeling tool, and includes multiple models to capture sensorimotor interactions with a task.

In this study, we built four models to model typical learning trajectories and outcomes in a declarative learning, LTM only system with a variable WM analog, a reinforcement learning system and combined RL, WM and LTM models. These models would allow us to test if the RLWM task can potentially be performed using declarative memory. Further, by exploring a range of parameters for learning rate (α), RL noise (τ), working memory (Imaginal-activation), memory retrieval noise and decay rate, we could estimate individual parameters and establish a link to the differences that amount to varied deployment of learning mechanisms.

Materials and Methods

Participants. 83 undergraduate students from the University of Washington participated in this experiment. All participants were monolingual English speakers recruited through the UW Psychology subject pool (47 females, aged 18-35 years). Data were collected after receiving informed consent in one 2-hour session.

Behavioral Task The Reinforcement Learning Working Memory task (Collins, 2018) involves learning stimulus-response associations through a series of 14 blocks. Participants are instructed to respond with a key-press of either 'C', 'V' or 'B' to the displayed images. In half the blocks, participants have to learn to associate key-presses with three unique images, presented 12 times in random order and in the other half with 6 unique images each presented 12 times within the block. The stimulus-response associations are deterministic and participants learn through reward (+1 point for correct responses and 0 points for incorrect responses). Following this learning phase, a 10-minute distractor task is administered before a surprise 206-trial test block. Participants make responses without feedback to items taken from both 3- and 6-set learning blocks. Stimulus presentations and data collection were done in MATLAB (mathworks.com).

Computational Models

All of the models experienced the same experimental set-up — 2 learning blocks of 3 and 6 objects respectively, a 10-minute break and a test phase without feedback.

Reinforcement Learning Model. The first model (Figure 1) most closely adheres to Collin's RL model. This model uses production rules to represent all of the possible stimulus-response associations,and uses reinforcement learning to progressively learn which associations are correct. Each production rule p has an associated *utility* value, $U(p)$, that reflects its expected rewards and is learned through a temporal difference rule. Specifically,

$$U_t(p) = U_{t-1}(p) + \alpha [R_t - U_{t-1}(p)] \quad (1)$$

in which α is the learning rate and R_t is the reward given at time t. In our experiment, R_t is binary and corresponds to the feedback ("Correct", R_t = 1, and "Incorrect", R_t = -1) given by the task interface. Competing responses are selected on the bases of their respective utilities, using a soft-max rule controlled by a noise parameter τ. The model initially responds randomly, until the correct rule accrues sufficient rewards to overcome the competitors, given the noise τ. The entire procedural/RL model is controlled by two parameters, the learning rate α and the selection noise τ.

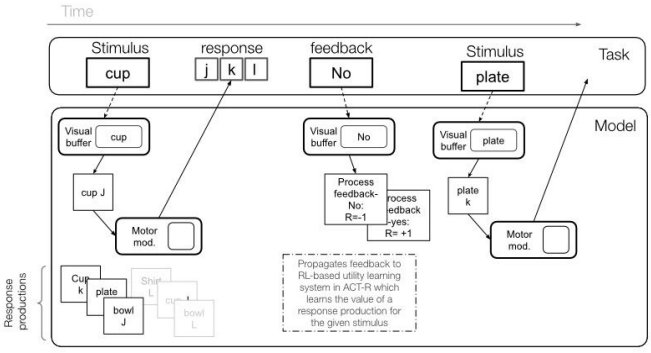


Figure 1: Overview of the procedural RL model, as implemented in ACT-R.

Declarative Learning Model. In lieu of Collins' pure WM model, we developed a *declarative* model (Figure 2), which manages both long-term and short-term explicit associations between a stimulus and its correct response This model stores memories of specific task events for later recall and use. To start, the model attempts to retrieve a memory of a previous response to the current stimulus that had resulted in a correct response. If such a memory is found, the same response is used. If no memory can be found, the model makes a random response. The outcome of this response to the current stimulus are then memorized. Although this model is computationally simple, ACT-R allows for a sophisticated control of the memory management processes through three parameters: (a) activation noise s, which captures random fluctuations in a memory's activations and associated probability of retrieval, (b) decay rate d, which captures the rate at which memories fade away and are forgotten (Sense et al., 2016); and (c) spreading activation weight W, which captures the attentional resources allocated to activating relevant memories during retrieval, and has been shown to capture individual differences in working memory capacity (Lovett, et al., 2000; Daily et al, 2001). We hypothesize that individual differences may occur in this three-parameter space and might be an intrinsic source of strategy choice during learning and retrieval.

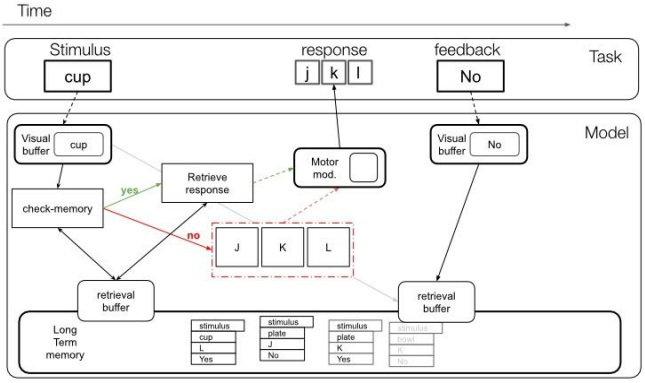


Figure 2: Overview of the declarative model, as implemented in ACT-R.

Integrated LTM-RL models. Our third and fourth models integrate the two single-system models into one model. Both models initiate each new trial by first deciding which of the two strategies to use---the procedural or the declarative strategy. The mechanism for integration provided a specific challenge. What is the most likely way that these two systems collaborate or compete during learning and recall? We decided to test two possible ways a meta-learner could arbitrate which system to use. The first, perhaps more elegant, solution was to have a reinforcement learner that learned the best strategy given the specific set of parameters. This model has five parameters total, the two inherited from the pure RL model (α and τ) and the three inherited from the Declarative model (s, d, and W). This model assumes that individuals are adaptive learners, and can optimally choose strategies based on their relative success over a short time. For example, if the long-term memory strategy proves too difficult (as in the case of too many stimuli), the model would switch to a RL-based learning strategy. RL learned associations are shared with the LTM system by inserting explicit information into the memory module.

The second integrated model has a built-in preference bias towards one system, quantified as a bias parameter β. Thus, at the beginning of every trial, the model selects the procedural/RL strategy with probability β and the declarative strategy with probability 1 - β. In contrast to the previous model, this bias is fixed and does not change over the course of the task. This model embeds the hypothesis that individuals might have established preferences towards one way to learn or another, perhaps honed over many years of "learning to learn" across

contexts and circumstances. For instance, if an individual has a preference for declarative learning, it would persist in trying to memorize stimulus-response associations even when switching to a RL strategy would be more convenient.

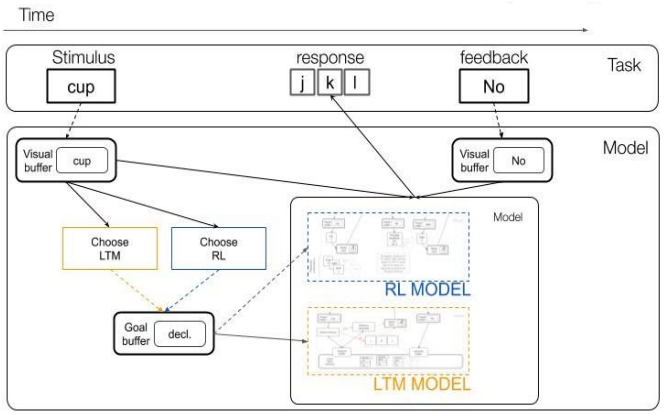


Figure 3: Overview of the two "integrated" models that employ both RL and declarative learning. The two models differ only in how they arbitrate between the two strategies.

Simulations

In this study, models are used as investigative tools to better characterize each individual. To do so, each model was run across a discretized version of its parameter space. Despite being computationally expensive and coarse, this method was preferred to convex optimization methods because it gives the full view of parameter space (including local and global minima) and, once computed, does not need to be recalculated for each participant. To obtain stable estimates, each model was run 100 times for each possible combination of parameters. In discretizing the range of each parameter, values were chosen to form an interval that surrounds the recommended value in the ACT-R documentation. A full description of parameters and the range of values that were manipulated is given in Table 1.

Table 1: Model parameters manipulated in the simulations

Parameter	Meaning	Values
α	Learning rate in RL	0.10, 0.15, 0.20
τ	Procedural rule selection noise	0.2, 0.3, 0.4
d	LTM decay rate	0.4, 0.5, 0.6
s	LTM activation noise	0.2, 0.3, 0.4
W	Spreading activation (Working memory capacity)	1, 2, 3

Data Analysis And Participant Fitting

Each participant's meta-learning strategy and latent, idiographic characteristics were then measured by identifying the model that best reproduced their observable data Y. Specifically, Each participant matched to a particular model M and set of parameter values θ_M, that minimized that following function:

$$M, \theta = \text{argmin BIC}(Y_p, Y_M \mid M, \theta)$$

in which Y_p is the observable task performance from participant p, Y_M is the simulated task performance, M is one of our four given models, θ_M is its associated set of parameters, and BIC is the Bayesian Information Criterion (Schwarz, 1978), which can be further expressed as:

$$\text{BIC} = n + n \log(2\pi) + n \log(\text{RSS})/n) + \log(n)(k+1)$$

in which n is the number of data points to fit, k is the number of parameters in each model, and RSS is the residual sums of squares. In our case, the n data points are the 24 means accuracies associated with the presentations of each individual stimulus (12 for Set3 and 12 for Set6), plus the two post-learning test accuracies.

The BIC was chosen because it incorporates both fit and model complexity in a Bayesian framework, thus natively accounting for the fact that a more complex model has an a priori greater likelihood to fit a given individual and that, given two models that fit equally well the same data, the one with the smallest number of parameters is the more likely to the be best model for that particular individual.

Results

Behavioral Results

By and large, our experimental results replicated the experimental findings of Collins (2018). This is shown in Figures 4 and 5, which illustrate the average performance of participants across the learning phase (Figure 4) and a comparison of the end of the learning phase vs. the test phase (Figure 5) of the task.

On average, participants' performance improved throughout the learning phase of the experiment, as shown by a significant effect of the stimulus repetition on its response accuracy [$F(11,984) = 405.67$ $p < 0.001$]. As previously reported, stimuli in Set3 condition was generally learned sooner and better than those of Set6. Finally, the two conditions interacted across learning and test phases [$F(1,328)$, $p < 0.01$], with learning for Set3 being more likely to decline from the end of the learning phase to the test phase.

As noted in Collins (2018), these group-level results strongly suggest that individuals use a mixture of declarative and procedural strategies. This is shown by the effects of the test phase (which suggest a decaying of information over time, possibly compatible with declarative memory) and by the superiority of the Set3 condition during learning (which rules out RL).

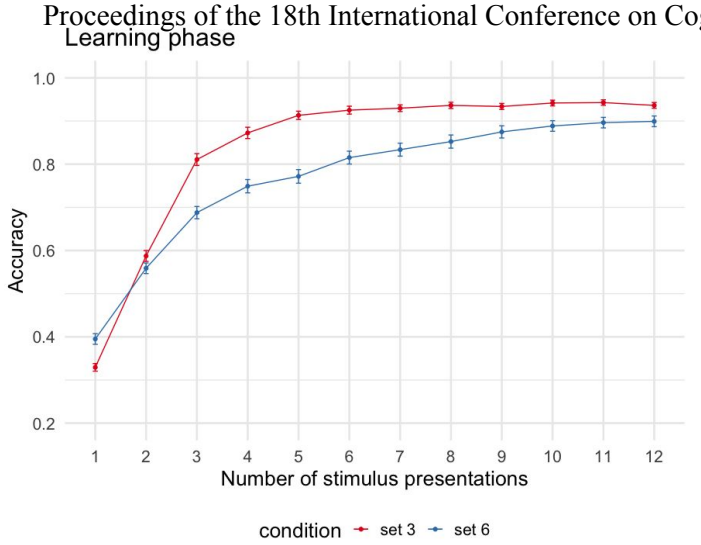


Figure 4: Accuracy across successive stimulus presentations during the RLWM task (Collins, 2018).

Figure 5: Accuracy during the test phase in the RLWM task (Collins, 2018).

Overview of Modeling Results

To give an idea of the general behavior of the four models, Figure 6 illustrates the mean performance of each of the four models during the learning phase. Although this data is averaged over all parameters and thus obscures the considerable variability across models (much like the group data in Figure 4 obscures the variability within subjects), it clarifies two important points. First, all of the four models, in general, capture the group-level learning rate. Second, even within the variability entailed by the different parameters, the models do predict different trends. As Collins (2018) pointed out, the pure RL model predicts no difference between Set3 and Set6. Notably, the pure LTM model also predicts no difference between the two sets, at least within our set of LTM parameters. The mixture models, however, do predict differences between the two conditions, with the difference being stronger for the explicit, biased meta-learning model. This is a side effect of the model using different strategies for Set3 and Set6 stimuli.

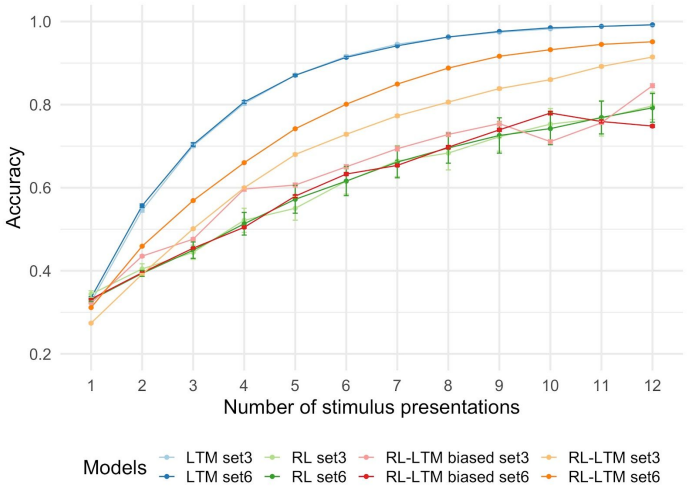


Figure 6: Learning trajectories for Set3 and Set6 stimuli for the four models.

Model Fitting Procedure

After examining the behavioral results, each participant was matched to an ideal model using the BIC criterion minimization procedure described above.

The results of this model fitting procedure yielded somewhat different results than the original study. We did not find that one model outperformed the others reliably. Rather, we found that different models steadily fit different subsets of participants (Figure 7). This was true even when, as in the case of integrated models, they effectively included the basic models as particular cases. In principle, this could be due to the fact that the BIC procedure does penalize more complex models.

Importantly, individual subgroups emerge even within the *integrated* models, suggesting that individual differences persist even at the level of meta-learning, or deciding which learning mechanisms to apply.

Conclusion

This study has used computational models to explore individual differences during learning. Specifically, this study has explored how different individuals engage alternative learning subsystems (declarative vs. procedural).

To do so, the study has capitalized on the use of idiographic computational models, that is, models designed to best fit a specific individual with a high degree of fidelity, rather than a group average—an approach that has recently gained prominence in cognitive neuroscience (Ceballos, Stocco, & Prat, 2020; Collins, 2018; Daw, 2011).

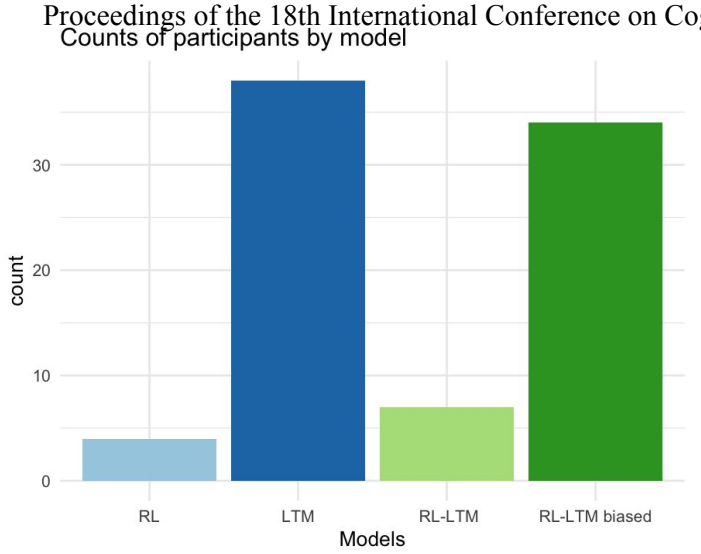


Figure 7: Count of the number of subjects that are best matched by each of the models.

It was found that different models fit different individuals, not only when models were effectively using different strategies (RL vs. declarative, LTM-based model) but also when one model was effectively nested within the other (basic models vs. integrated models). More importantly, it was found that the principle that different individuals fit different models also applies to higher-level models. In our case, the two "integrated" models were found to better fit different participants, with some adapting their learning strategy during the task, and some maintaining a bias towards one learning system. To the best of our knowledge, this is the first study to report such findings.

A number of limitations must be acknowledged. First, the number of models we explored is still limited. Second, and most importantly, the size of the parameter space that was explored was extremely small. Both of these limitations will need to be overcome in future research and are currently limited by computing power. We are leveraging the use of cloud computing, as suggested by one of our reviewers, to search a wider range of parameter values. This will also afford us better fit between our models and behavioral data and parameter estimation than we have currently achieved.

These limitations notwithstanding, a number of important points need to be made. The first is that individual differences do matter and, as it is becoming increasingly apparent, group data might not reflect the true behavior of any of its component individuals. Computational models provide a new and unique method to understand, measure, and uncover the dimensions in which individuals differ from one another.

A second, point to be made concerns the importance of declarative memory in learning strategy, at least in humans, even in its long-term form. The success and prominence of RL theory in neuroscience has led to probably overlooking how much individuals rely on declarative strategies in learning simple response associations tasks. This is apparent in Collins' (2018) and Collins and Frank's (2012) conclusions, which, while acknowledging working memory, dismiss the possibility of participants forming long-term declarative associations altogether. Instead, our modeling results suggest that declarative-based models fit large sub-groups of individuals. Even the simplest, non-integrated model, accounts for 36% of our participants, and, altogether, models that at least include declarative components account for 73 out of 83 participants (Figure 7). Our results are also consistent with the increasing popularity of declarative memory-based approaches to learning and decision-making, such as the popular decision-by sampling (Stewart, Chater, & Brown, 2006) and Instance-Based Learning (Gonzalez, Lerch, & Lebiere, 2003).

A third and related point that needs to be made is that, while models do matter, the specific type of modeling approach that is used matters even more. It would have not escaped the attentive reader that, while our empirical results largely mirror those of Collins (2018), our conclusions do not. This is mostly due to the fact that our choice of modeling paradigms was different, and carries different assumptions about the cognitive system. Consider the difference in learning between Set3 and Set4 conditions. Collins' (2018) explanation is that Set3 items are more likely to be still in working memory during learning, thus facilitating performance by direct reading of the associated response from a short-term buffer. Our explanation is that participants probably relied on different learning systems LTM vs. RL for the two sets of stimuli. Because the space of possible models is so large, it is practically impossible to empirically decide on this matter. For this reasons, we advocate for developing idiographic (i.e., individual-level) models within an integrated cognitive architecture, so that the different models are more clearly comparable and benefit from a common, well established set of constraints (which seems to be evolving towards a consensus: Laird, Lebiere, & Rosenbloom, 2017). By doing so, we believe we have put this research on a better footing for future developments.

References

Anderson, J. R. (2007). How can the human mind occur in the physical universe? *Oxford University Press*.

Atkinson, R.C.; Shiffrin, R.M. (1968). Human memory: A proposed system and its control processes. In Spence, K.W.; Spence, J.T. (eds.). *The psychology of learning and motivation. 2*. New York: Academic Press. pp. 89–195.

Ceballos, J. M., Stocco, A., & Prat, C. S. (2020). The Role of Basal Ganglia Reinforcement Learning in Lexical Ambiguity Resolution. *Topics in Cognitive Science, 12*(1), 402-416.

Collins, A. G. (2018). The tortoise and the hare: Interactions between reinforcement learning and working memory. *Journal of cognitive neuroscience, 30*(10), 1422-1432.

Collins, A. G., & Frank, M. J. (2012). How much of reinforcement learning is working memory, not reinforcement learning? A behavioral, computational, and neurogenetic analysis. *European Journal of Neuroscience*, *35*(7), 1024-1035.

Daily, L. Z., Lovett, M. C., & Reder, L. M. (2001). Modeling individual differences in working memory performance: A source activation account. *Cognitive Science, 25*(3), 315-353.

Daw, N. D. (2011). Trial-by-trial data analysis using computational models. *Decision making, affect, and learning: Attention and performance XXIII*, 23(1).

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, 27(4), 591-635.

Laird, J. E., Lebiere, C., & Rosenbloom, P. S. (2017). A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics. *AI Magazine, 38*(4), 13-26.

Lovett, M. C., Daily, L. Z., & Reder, L. M. (2000). A source activation theory of working memory: Cross-task prediction of performance in ACT-R. *Cognitive Systems Research*, 1(2), 99-118.

Kane, M. J., Bleckley, M. K., Conway, A. R., & Engle, R. W. (2001). A controlled-attention view of working-memory capacity. *Journal of experimental psychology: General*, 130(2), 169.

Miller, E. K., Lundqvist, M., & Bastos, A. M. (2018). Working Memory 2.0. *Neuron*, 100(2), 463-475.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2), 461-464.

Stewart, N., Chater, N., & Brown, G. D. (2006). Decision by sampling. *Cognitive Psychology*, 53(1), 1-26.

Stocco, A., Lebiere, C., & Anderson, J. R. (2010). Conditional routing of information to the cortex: A model of the basal ganglia's role in cognitive coordination. *Psychological review*, 117(2), 541.

An Imperative Alternative to Productions for ACT-R

Anthony M Harrison (anthony.harrison@nrl.navy.mil)
U.S. Naval Research Laboratory
Washington, DC, 20375 USA

Abstract

As cognitive modeling has matured, so too have its tools. High-level languages are such tools and present a rich opportunity for the acceleration and simplification of model development. Reviewing some of a the major contributors to this area, a new language (Jass) is introduced for building ACT-R models. Jass simplifies and accelerates model development by providing an imperative language that is compiled to production rules. A complex model implemented using this language is detailed.

Keywords: cognitive modeling, ACT-R, high-level languages, cognitive architectures

Introduction

Cognitive modeling allows us to generate high-fidelity models of human behavior for a wide array of applications and research questions. These models are the result of complex, time-consuming development processes. Many researchers and engineers have developed tools and theories in order to simplify, democratize, or otherwise accelerate this development process (e.g., John, Prevas, Salvucci, & Koedinger, 2004). Of particular interest at present is the work on higher-level languages for cognitive modeling. These languages all share a common goal of simplifying the process of modeling by abstracting away some set of low-level features, while retaining or extending functionality. The theoretic commitments and approaches vary, but common across most of the studies in recent years has been the commitment to a *compilation model* (Ritter et al., 2006). In a *compilation model* high-level source code is compiled into production rules to be run on the underlying cognitive architecture. This permits models developed in the high-level language to avoid low-level pain points while still having access to the full explanatory power of the underlying architecture.

To differentiate the various approaches to the compilation model it is worth considering the intended scope of the generated models and the degree of theoretic commitment the language makes. Degree of commitment is the extent to which the language makes modeling commitments for the modeler. For instance, some of these languages provide a *press-button* construct for computer interaction. While the construct is simple enough, it can actually be implemented in many different ways at the lowest level. The language could employ a single implementation, theoretically committing the user of the language to that approach. Alternatively, the language could provide a set of implementations, allowing the user to select a particular commitment. Yet another alternative is to minimize the commitment by not providing the construct at all, leaving the implementation entirely up to the user. These design decisions make theoretic commitments that can obstruct or hinder some theoretic accounts. Generally speaking, the higher the level of abstraction the greater the number and degree of theoretic commitments made. More precisely, all of these languages commit the modeler to a specific goal structure and management which would make it difficult to study alternative accounts for those elements.

The earliest high-level compiled language for ACT-R was ACT-Simple (Salvucci & Lee, 2003). This GOMS-like language took basic primitives (e.g., click, type, speak, think) and mapped them to a fixed set of specific production sequences. Intended for simple, serial, computer-based tasks, the executable models are able to make realistic time-based predictions. The basic primitives used in ACT-Simple are appealing from a user-design evaluation perspective but they are too limiting for the modeling of general tasks. GOMS is an abstract analysis formalism, not a programming language proper, and therefor lacks many of the constructs necessary for general modeling tasks (e.g., error handling). As described in the earlier *press-button* example, ACT-Simple used fixed mappings between behaviors (e.g., *press-button*) and production sequences, committing the user to those particular theoretic accounts.

GOMS to ACT-R (G2A) (St Amant, Freed, & Ritter, 2005) adopts a similar approach as ACT-Simple. Also based on GOMS, G2A inherits some of the challenges therein. It simply lacks many of the concrete formalisms necessary to express arbitrary task and flow structures. While still well suited for simple, computer-based tasks, G2A does vary the primitive-production mapping. Recognizing that multiple production solutions exist for each primitive, G2A allows those mappings to be manipulated at compile time. This low degree of commitment permits G2A to generate families of related models that vary in individual theoretic commitments. It is unclear if the user of the language is able to contribute these primitive-production mappings directly.

The High Level Symbolic Representation (HLSR) language (Jones, Crossman, Lebiere, & Best, 2006) shares many features with Prolog, and brings with it the rich functionality of that programming language. What sets HLSR apart is that the compiler is able to target multiple cognitive architectures, generating productions for both ACT-R and SOAR. This opens the door to performing architectural comparisons while holding the model itself fixed. At such a high-level of abstraction, HLSR has a high degree of commitment, which while enabling greater productivity, ultimately limits the language's ability to provide alternative accounts.

Herbal (Paik, Kim, & Ritter, 2009; Paik et al., 2010) is a graphical language, based on Newell's Problem Space Computational Model (Newell, Yost, Laird, Rosenbloom, & Altmann, 1993). Designed initially for SOAR, it was adapted

to also compile to multiple cognitive architectures, including ACT-R. Similar to HLSR, Herbal is intended as a general modeling language, even though it makes significant theoretical commitments. Herbal's most novel contribution is in the generation of productions at multiple levels of competency. Its production generator can output novice models that rely predominantly upon declarative representations and a few general purpose productions; full expert models where the declarative components have been fully proceduralized; and any mixture of the two endpoints. In order to provide this feature, Herbal commits the user to very specific goal structures and management, as well as specific declarative representations for instructions.

This paper introduces the jACT-R Assembler (Jass)[1], a high-level imperative language that compiles into ACT-R productions for the Java implementation of ACT-R, jACT-R [2]. It aims to simplify development by abstracting away low-level productions, while retaining the full range of ACT-R's explanatory power.

jACT-R Assembler (Jass)

Motivation

Jass[3] is intended for modelers specifically familiar with the theoretic underpinnings of ACT-R. It codifies ACT-R's theoretic constraints while providing significant control over how a model's commitments are expressed. The language was developed with three goals in mind: eliminate the production as the unit of development, maximize the modeler's theoretic control by minimizing the language's commitments, and to allow the language to co-evolve with ACT-R.

Eliminate the Production

Central to this work is the thesis that developing in production rules is itself a hindrance to developing cognitive models. Because of the small granularity of productions, it is often challenging to understand what any one is doing without understanding those that are supposed to fire before and after it as well. This implicit ordering problem makes understanding and developing productions very difficult. Any cognitive resources that can be freed up by not working on productions can be directed towards the development of the model's core theoretic predictions. However, productions lie at the heart of ACT-R as a theory and cognitive architecture. Jass achieves this abstraction by providing an imperative C-style language from which productions are generated for execution by the architecture (see Figure 1).

Production Generation Unlike Herbal (Paik et al., 2010), Jass does not to produce novice or expert models, rather something in between, an intermediate model where all the declarative information has been compiled out, but the sequencing and timings have yet to be optimized. The productions generated by Jass are also sparse in that there is at most one instruction per production. This is in contrast to normal, dense, hand-coded productions which often load multiple instructions into a single production. Early work suggest that Jass models have around 3x more productions than hand-coded models. Productions generated by Jass still have room for compilation and optimization by the architecture's production compilation mechanism (Taatgen & Anderson, 2002).

```
/*
 * See something? press button
 */
function void TaskA() {
  slot tmp = null
  //reset the visual system
  request visual(resetVisual)
    =>{
    tmp=null
  }
  // wait for something to be seen
  while(goal(tmpIsNull))
  {
    request visual-location(newVisualLocation)
      -> tmp = visual-location
      =>{
      tmp=null
    }
  }
 // and press a button
  request motor(buttonPress)
    => {
    tmp = null
  }
  return
}
```

Figure 1: Simple perceptual-motor task in Jass.

Productions for Goal Management At the heart of all ACT-R models are the productions that manage the current goal. A random sampling of available published models [4] shows that the vast majority of models (90%) use an explicit state representation to control production flow. That is, the goal has a single, perfectly predictive variable devoted to controlling production sequencing, as opposed to controlling sequencing using multiple variables or states. Given that fact, Jass adopts a similar goal structure, with an explicit state variable. However, because goal management is still an area of active research, management is implemented as a pluggable interface. This allows the goal management to be swapped out as necessary so long as there exists a variable devoted to maintaining the explicit state. Jass subsumes goal manage-

[1] https://github.com/amharrison/jass

[2] http://jact-r.org/

[3] Jass is implemented using Eclipse's Xtext language development toolkit. http://eclipse.org/xtext/

[4] http://act-r.psy.cmu.edu/publication/, as of 2/20/20

ment for the modeler by transforming goal management into imperative function-calls. That is, each goal is expressed as a callable function with parameters. In this way, the modeler is freed from managing goal representations themselves and instead just structure function calls to complete the actual goal. Without productions, the language is effectively one of managing the contents of the working memory buffers.

Theoretic Control

Theoretic control is the ability to express a particular modeled behavior using the full extent of the theoretic framework. Tools with a high degree of theoretic commitment limit one's theoretic control. The aim of Jass was a language designed around the architecture as it is used, hopefully enabling it to avoid many of the theoretical commitments present in prior work. By designing around the architecture we can support or fully codify the five major modeling paradigms in ACT-R (Taatgen, Lebiere, & Anderson, 2006). This allows Jass to exploit ACT-R's full theoretic coverage and not just a subset of it. Two of these paradigms, instance learning and competing strategies, are so pervasive in ACT-R models that they were reified as language constructs.

Competing Strategies Production rules are heavily parallel by their very nature. But since ACT-R imposes a serial processing bottleneck, only one of many competing productions can be selected for firing at any given time. This very process accounts for many modeled phenomena in ACT-R. But because of low-level of productions it can be difficult on casual inspection to determine which productions are supposed compete without running the model directly. Jass makes this explicit through the use of the *match-case* statement. Normally Jass's generated productions are strictly serial, following the imperative instruction order. When it reaches the *match-case* statement, it knows to generate a competing production branch for each *case* encountered. Figure 2 shows a snippet that picks between three strategies and falls back to *default* in case none match the current system state.

Buffer Requests Instance learning in ACT-R is the retrieval and application of prior problem or goal state information. To achieve this, productions must make a request of the retrieval buffer to fetch from declarative memory some matching pattern (see Figure 3). The pattern of requesting and using information from a particular buffer is so pervasive that most major predictions are derived from the consequences of these requests. As such all request patterns are subsumed by Jass's *request* statement. The behavior of the *request* instruction is determined by the buffer that it is making the request of. Jass includes a contributable meta-definition for buffers that defines their expected behavior. For instance, a buffer can be marked as having a potential error state which will require the *request* instruction to have an error handler. Figure 3 shows a retrieval request (7) of something matching *underspecifiedEpisode* with success and error handlers.

```
...
match{
  case goal(nextIsA) : {
    TaskA()
  }
  case goal(nextIsB) : {
    TaskB()
  }
  case goal(nextIsC) : {
    TaskC()
  }
  case goal(nextIsD) : {
    TaskD()
  }
  default : {
     TaskA()
 }
```

Figure 2: Match-case statement with three alternative branches competing with the default branch. Priorities can be specified using [#] after the case. Function calls denote a change of goal.

Commitments The greatest commitment that Jass makes is to the mapping of language constructs to production rules generated. Implemented as a pluggable, extensible interface, new mappings can be swapped in or added if the current commitments are deemed inadequate. The next major commitment is to the goal structure, but as previously mentioned it should be able to handle the majority of models. It too is implemented as a pluggable interface should the goal commitments need to be modified.

Evolve with Architecture

Cognitive architectures are implementations of evolving theories. To be truly useful, any high-level language must be able to evolve with its underlying architecture. Failing to do so ultimately undermines the tools influence and utility (Ritter et al., 2006). As mentioned previously, Jass uses a pluggable software architecture for all of its major components. This allows goal management and even individual language constructs to be swapped out as theoretical explorations dictate. The language also directly supports the contribution of new modules and buffers through the buffer meta-descriptor (Figure 4). This makes it possible to consolidate the various buffer behaviors into the singular buffer *request* construct discussed earlier.

Memory for Goals: A Test Case

To gauge the relative success at achieving the design goals of Jass a validation model was implemented. That model should in someway inform each of the design goals discussed previously. Specifically, the elimination of the production should facilitate the development of more complex models; a high

```
underspecifiedEpisode = {
  isa episode
}

...

request retrieval(underspecifiedEpisode)
  -> current = retrieval.reference
  =>{
      ... //error handler
    }
...
}
```

Figure 3: *Request* of retrieval module to fetch a chunk matching *underspecifiedEpisode*. On success, grab the reference. On failure, do something else.

```
goal writable requests * -> *
imaginal writable requests * -> *
retrieval readable requests * -> *
motor error requests motor-command -> ,
                     motor-clear ->
visual readable error
          requests move-attention ->
                   visual-object ,
          clear ->
```

Figure 4: Jass's buffer meta-descriptor specifying writability, potential for error, and the expectations of the *request* statement.

degree of theoretic control should allow the modeler fully exploit the underlying architecture; and it should adapt new theoretic contributions seamlessly. Altmann & Trafton's Memory for Goals models (2007; 2011) fit these requirements.

Memory for Goals

Memory for goals is a theory of goal management extensively applied to interruptions that accounts for various resumption errors (Trafton et al., 2011) and lags (Altmann & Trafton, 2007). It posits that we rely upon short-lived episodic traces and their retrieval to manage our goals. Resumption errors are due to the inappropriate retrieval of noisy episodes, and lags are due to incrementally rebuilding episodic context after interruption. These features map nicely to the design goals of Jass. First, the model is complex requiring multiple tasks and their interleaving due to interruption. Second, it relies upon unanticipated uses of ACT-R's underlying architecture (i.e., clearing the goal buffer for an interruption) and makes strong predictions about goal usage. Finally, their account makes use of custom episodic module (i.e., a novel theoretic contribution outside of Jass's initial design scope).

Experimental Task The original primary task was a complex computer game (Trafton, Altmann, Brock, & Mintz, 2003) that exhibited two primary features. First, the frequency of response was high permitting the collection of numerous samples as recovery interruption recovery progresses. Second, the task is complex enough that it requires some cognitive state for an interruption to disrupt. The interrupting task was a radar-classification task (e.g., Brock, Stroup, & Ballas, 2002) where subjects selected targets and classified them based on simple rules. For every twenty minute block of the primary task, there were twelve randomly distributed interruption phases. Reaction times were recorded for the first ten responses after an interruption resumption. Each participant completed three blocks (early, middle, late) to assess learning. The remaining details can be found in (Altmann & Trafton, 2007).

Model

Since this modeling endeavour was more of a proof-of-concept than a rigorously validated model, large portions of the primary and interruption tasks were simplified, focusing primarily on the core of memory for goals.

Modeled Tasks Three independent Jass libraries were developed, one for each of the interruption, primary, and management tasks. The interruption task was modeled as an exhaustive visual search, followed by some key inputs. The primary task was itself made up of multiple smaller Jass libraries, each designed to be basic perceptual/action tasks strung into a repeating sequence. It was the manager task's job to determine which of the primary tasks to run at any given time. During normal execution, the model alternates between the manager and the next primary task to be executed. On interruption, the working memory buffers were cleared, triggering the interruption task.

The manager encapsulates the majority of Memory for Goal's theoretic account. Under normal conditions, the manager tracks the prior and current tasks. This context allows it to rapidly retrieve the next task in the sequence. Under interrupted conditions, this context is wiped out and the task manager must try to retrieve the most recent episode which contains the tag representing the completed task. Assuming the task was completed, the task manager tries to retrieve the next task in the sequence. With the to-be-completed task known, the manager creates a new episodic encoding and optionally rehearses it, if it is currently rebuilding context. This new episode is then retrieved, relying only upon the spreading activation from the current context for priming. The retrieved episode (possibly incorrect due to noise) is then used to execute the next task.

Goal Management While memory for goals makes specific predictions regarding how goals are rehearsed and retrieved for execution, it is silent on the actual form of the goal. Because of this, Jass's default goal management was able to be used without any modification. However, Altmann

& Trafton's use of buffer clearing to model interruptions did require the inclusion of three hand-written productions to deal with the empty goal buffer state.

Episodic Module Memory for goals depends upon some form of an episodic module. Altmann & Trafton underspecify this component, choosing instead for a minimal commitment. All this episodic module does is create a unique, time-stamped, chunk with a single reference. This reference can be to anything and in their models it is the task representation of the to-be-completed task. Jass was easily able to accomodate the new module using the buffer meta-descriptions mentioned previously (see Figure 4).

Results

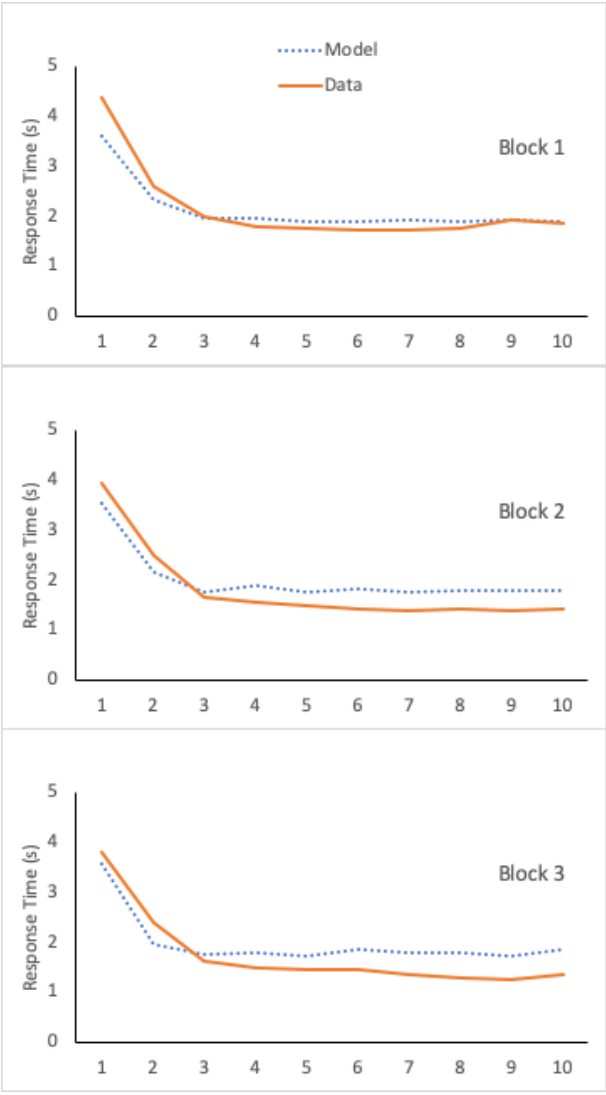


Figure 5: Average response times from the Altmann & Trafton (2007) experiment (solid lines) and average model fits (dotted lines), plotted by block (1-3) and serial position after interruption (1-10).

Model The model was run one hundred times with an activation noise of 0.1, all other parameters were set at their defaults. Model and empirical response times are plotted in Figure 5. The model fits well ($RMSE = 0.243$, $R^2 = 0.94$). This shows the primary resumption lag effect as it rebuilds its context after interruption. Unfortunately, the model does not show the same learning effect across blocks as seen in the empirical data. This is due to the relatively lean declarative needs of the model, that is, only the episodes and task tags are retrieved. Had there been task information retrieved during the execution of the primary and interruption tasks, we'd expect to see a greater effect of practice.

While the general pattern of the reaction times is consistent with the empirical findings, the model is consistently slower at the later positions. At this late point, declarative retrievals are effectively immediate, the latency is largely due to the overhead of the productions. Had production compilation been enabled, we'd expect the generated production overhead to be reduced, making up the difference.

Jass While too early for a formal study, it is worth considering the anecdotal experience of modeling memory for goals in Jass. From inception to first batch runs was less than four days of engineering time. The coding of the three tasks took less than eight hours, yielding a moderately sized model of 183 productions, approximately 22.5 productions per hour. In terms of lines of code, the Jass models took a combined 592 lines versus the generated productions taking a combined 2945 lines (5x more compact).

Discussion

We successfully demonstrate the use of Jass to develop complex cognitive models for ACT-R. The imperative model simplifies the temporal sequencing of actions required for task completion relative to working with productions directly. The design of Jass allows it to accommodate many different theoretical accounts, even for core elements such as goal management. The design flexibility also permits Jass to adapt to changes in the underlying architecture, allowing it to keep abreast of current theoretical trends. Jass's compilation mechanism effectively creates goal-based libraries of functionality. Each of the modeled tasks was implemented separately and only combined into a single model at run time. This is a promising feature as it applies to model reuse across projects and researchers. As a tool for cognitive modeling, this simple proof-of-concept bodes well for the utility of Jass. However, much more rigorous usability testing is required to get a full sense of the tool's benefits and drawbacks (Ritter et al., 2006).

Acknowledgments

This work was supported by ONR under funding document N0001420WX00496 awarded to Dr. Laura Hiatt. The views and conclusions contained in this document should not be interpreted as necessarily representing the official policies of the U.S. Navy.

References

Altmann, E. M., & Trafton, J. G. (2007). Timecourse of recovery from task interruption: Data and a model. *Psychonomic Bulletin & Review*, *14*(6), 1079–1084.

Brock, D., Stroup, J. L., & Ballas, J. A. (2002). Effects of 3d auditory display on dual task performance in a simulated multiscreen watchstation environment. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 46, pp. 1570–1573).

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 455–462).

Jones, R. M., Crossman, J., Lebiere, C., & Best, B. J. (2006). An abstract language for cognitive modeling. In *Proceedings of the 7th iccm.*

Newell, A., Yost, G. R., Laird, J. E., Rosenbloom, P. S., & Altmann, E. (1993). Formulating the problem space computational model. In *The soar papers (vol. ii) research on integrated intelligence* (pp. 1321–1359).

Paik, J., Kim, J. W., & Ritter, F. E. (2009). A preliminary act-r compiler in herbal. In *Proceedings of iccm-2009-ninth international conference on cognitive modeling* (pp. 466–467).

Paik, J., Kim, J. W., Ritter, F. E., Morgan, J. H., Haynes, S. R., & Cohen, M. A. (2010). Building large learning models with herbal. In *Proceedings of iccm-2010-tenth international conference on cognitive modeling* (pp. 187–192).

Ritter, F. E., Haynes, S. R., Cohen, M., Howes, A., John, B., Best, B., ... Lewis, R. L. (2006). *High-level behavior representation languages revisited* (Tech. Rep.). PENNSYLVANIA STATE UNIV STATE COLLEGE COLL OF INFORMATION SCIENCES AND ....

Salvucci, D. D., & Lee, F. J. (2003). Simple cognitive modeling in a complex cognitive architecture. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 265–272).

St Amant, R., Freed, A. R., & Ritter, F. E. (2005). Specifying act-r models of user interaction with a goms language. *Cognitive Systems Research*, *6*(1), 71–88.

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say "broke"? a model of learning the past tense without feedback. *Cognition*, *86*(2), 123–155.

Taatgen, N. A., Lebiere, C., & Anderson, J. R. (2006). Modeling paradigms in act-r. *Cognition and multi-agent interaction: From cognitive modeling to social simulation*, 29–52.

Trafton, J. G., Altmann, E. M., Brock, D. P., & Mintz, F. E. (2003). Preparing to resume an interrupted task: Effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human-Computer Studies*, *58*(5), 583–603.

Trafton, J. G., Altmann, E. M., & Ratwani, R. M. (2011). A memory for goals model of sequence errors. *Cognitive Systems Research*, *12*(2), 134–143.

Ethical Test Driven Development: A Design Process for Building Ethical Agents

Steve Highstead
Carleton University
Ottawa, Canada
(stevehighstead@cmail.carleton.ca)

Robert West
Carleton University
Ottawa, Canada
(robert_west@carleton.ca)

Jennifer Schellinck
Carleton University
Ottawa, Canada
(jschellinck@sysabee.com)

Babak Esfandiari
Carleton University
Ottawa, Canada
(babak@sce.carleton.ca)

Abstract

As machines become autonomous, acting as agents within society, there will become an increasing need for them to interact with people. For a machine to act within a society free of its creator's supervision, it will also have to have the same capacity for intersubjective behavior as people. This paper presents a design system for creating an artificial moral agent based on cognitive modeling using test-driven development.

Keywords: machine morality; artificial agents; moral dilemma; test-driven development; autonomous machines

Introduction

Machines that can act autonomously are becoming ubiquitous in human society. Included are machines such as autonomous vehicles (Lin, 2013), care attendants on Alzheimer wards (Anderson & Anderson, 2007), in-home care givers, and customer attendants (Bekey, 2012). These machines will operate within society without human intervention and as such need to be programmed to make goal based judgements, not only with respect to what action to take, but also with respect to how the actions will be executed. Because they are interacting with people, they will be required to do so in a manner that is acceptable to human beings, much in the same way that people currently interact with each other.

The area of discourse that deals with acceptable intersubjective behavior is ethics. Since autonomous machines will be interacting with people, they will need to behave in a way that is morally acceptable (Bonnefon, Shariff, & Rahwan, 2016; Anderson & Anderson, 2007; Allen & Wallach, 2009; Lin, Abney, & Bekey, 2011).

Ethics has a long and varied history. However, there are no settled set of universal rules for moral behavior (Moor, 2006). Ethics discourse is the realm of thinking that tries to understand how human behavior can be morally evaluated. For an engineer or a programmer, though, it can be problematic to translate this into something that is functionally useful (Allen & Wallach, 2009). At the same time, the interests of an engineer or programmer, are problematic for an ethicist to appreciate. What is needed is a mechanism to bridge these two realms. To this end, we propose a software development process for building artificial ethical systems.

Artificial Moral Agents

An agent is any entity, artificial or human, that has the capacity to sense, formulate intentions, and plans to act upon its environment. For Bratman, an agent can act purposively, and has the capacity to form and execute plans. (1987). An agent can sense, assess and evaluate, and possesses the ability to act or not upon matters of fact within an environment. From a cognitive science perspective, "a rational agent is one that can critically reflect upon her reasons for action and come to a deliberative conclusion about what she ought to do" (Rini, 2015). Wooldridge defines an artificial agent as "a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objective" (2009).

If the artificial agent is operating within a human society, and if its purpose does not take into account its interactions with people, then it can unwittingly imperil humans. As with human agents, an artificial agent requires an ability to know what is and is not acceptable behavior. To achieve this, the tasks that fulfill the artificial agent's purpose require ethical guidance, much like human agents.

An artificial agent that possesses additional functionality that governs and ameliorates its actions with regard to other agents is what Wallach refers to as an *artificial moral agent* (Allen & Wallach, 2009). James H. Moor along with Judith Leigh Anderson and Michael Anderson refer to such agents as *explicit* ethical agents (Anderson & Anderson, 2007; Moor, 2016). This type of agent has an explicit ethical feedback system that monitors and judges planned actions. This has also been referred to as an "ethical governor" (Arkin, Ulam, & Wagner, 2012) or an "ethical layer" (Vanderelst & Winfield, 2017).

Explicit ethical agents can be contrasted with *implicit* moral agents. An implicit moral agent does not have a distinct set of moral functions that provides feedback on planned actions. Instead, for this type of agent, ethical behavior is considered as an integrated part of the task and coded as such. Morality, in this type of system, is a type of situated action

(Moor, 2016), rather than a principled or rule-based judgement.

Design Process

Cognitive modeling is uniquely situated to create ethical artificial agents and has contributed important insights on how to build human like moral agents (Wallach, 2010). However, these insights, similar to the insights of Moral Philosophers, need to be implemented within a design process in order to create agents that can operate in the real world, who have the potential to harm or help human beings.

Broadly speaking, there are two design approaches to implementing an ethical framework in an artificial agent, "top down" and "bottom up" (Allen, Smit, & Wallach, 2005) In a top down methodology, a set of rules are programmed into a machine (Saptawijaya & Pereira, 2016). They can be as basic as Asimov's three laws, (1950) or as involved as a full code of ethics. In a bottom up methodology, the artificial agent learns the rules as it encounters ethical dilemmas and receives ethical feedback on the choices it makes, not unlike parental moral guidance for intelligent machines, (Rini, 2017). The top down approach can be used to create either explicit or implicit agents, while the bottom up approach is more naturally aligned with creating implicit moral agents.

In the top down approach, identifying rules of behavior is in and of itself problematic. Even with as few as Asimov's three rules, conflicts arise, which the author employed as rich plot devices for his stories (1950). Using this methodology, it is challenging to identify all possible scenarios for which an applicable rule would suffice. However, if the environment is very restrictive, using this approach can produce viable results (Vanderelst & Winfield, 2017).

A bottom up agent (such as a deep leaning network) could potentially avoid these problems. However, if these systems are overfit to the learning set and/or the learning set is missing some scenarios, they can make unpredictable decisions when the situation deviates from the training set, which is worrisome when human lives are on the line. One interesting approach is to build hybrid systems that are both bottom up and top down such as the Clarion architecture (Sun, 2007).

Top-Down Design

Creating an ethical agent from a philosophical starting point can be understood in terms of what philosophers refer to as an ideal observer. According to Firth, (1952), an ideal observer has perfect knowledge of non-moral facts, perfect knowledge of the situation, and is logically consistent. Ethical judgements then emerge from these pre-existing conditions. Effectively, the ideal observer is a model of a perfect but disembodied moral agent. Starting from this point the goal would be to solve the problems of perception, action, and embodiment so that the disembodied moral agent can act in the world.

However, the ideal observer leads to a problematic software development process. Intuitively, it feels like the division of labor should involve philosophers first developing ideal observers and then passing the requirements to programmers and engineers to solve the embodiment problem. This development process is effectively, the waterfall software development process, or Waterfall Method.

The problem with the waterfall method, is that it assumes the abstract principles at the top will cover any and all real world issues satisfactorily. This is problematic. Even if a set of ethical principles is sufficient, which is unlikely, it does not tell us how to ground or embody those principles for real-world effectiveness. Also, because it is top down, the design process is biased toward the creation of explicit agents. Finally, there is no explicit space in this design process for cognitive modeling. It proceeds straight from philosophy to engineering. To offer an alternative, one with cognitive modeling in the loop we developed a software development system based on test-driven development.

Bottom-Up Development

The methodology of Test-Driven Development (TDD) was formalized by Kent Beck in his book *Test-Driven Development by Example* (2003). Subsequently there have been additional resources that have become available such as David Astels' *Test-Driven Development: A Practical Guide* (2003). Philosophically, test-driven development follows the Popperian notion of falsification. The idea is initially to determine a test for software, before any software is written. First, software specifications for functionality and features are formulated as a test. The test is then performed on the software. If the test fails, which it should initially since there is no software code that implements what is being tested, software is written and tested until such time the test is passed. The software code can then be refactored and cleaned to remove any duplication or inefficiencies. Once the test is passed, the development cycle begins once more with the addition of another test (Figure 1). On each round the code must pass all the previous tests.

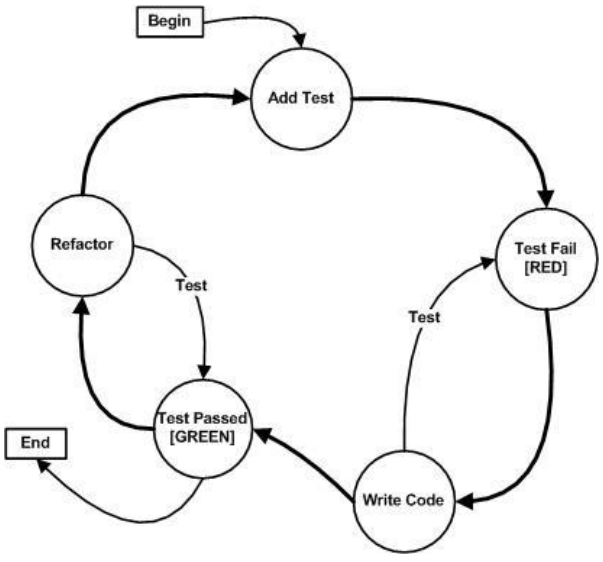


Figure 1 Test-driven Development Cycle

In our system, ethical concepts developed within the ideal observer must be translated into specific tests. These tests are stored in a component called the Oracle. That is, philosophers must operationalize the ideal observer to generate specific moral tests. The programmer can see only the tests in the Oracle and must interpret them in terms of the agent and the environment. While conceptually, philosophers are the ideal source of ethical wisdom, there is nothing that precludes other sources for the Oracle such as AI search algorithms.

However, feedback is also important. Programmers need to be able to feedback questions to the philosophers about tests that are not specified sufficiently to translate into the TDD environment. Also, the philosophers need to see the results of the tests to check for any unforeseen consequences that fall outside of the specific tests. This is illustrated in Bostrom's example of an AI for making paper clips. The AI's purpose is to make paper clips using available resources. Humanity becomes just another resource, which, of course we find ethically repugnant (2014). This is an extreme example, but it illustrates the point. Embedding the tests in realistic simulations or even real-world situations, is critical for detecting unintended consequences.

Since it is a continuous development process, much like human agents continuously learning and solving ethical dilemmas, the software agent will gradually develop a body of ethical knowledge consistent with its operational environment. Importantly, ethical test-driven development can be integrated with regular test-driven development so that the ethics of the agent is never decoupled from the abilities of the agent.

Methodological Demonstration

To demonstrate this methodology, a software model was developed to emulate the classic Trolley Problem (Foot P. , 1967; Thompson, 2009). Thompson presents the two most common scenarios as follows:

1. A trolley has lost its brakes and cannot stop. There are five people on the track out of sight of the trolley and they cannot get off the track in time before the trolley hits and kills them. The track has a spur line onto which the trolley can be switched, but there is also one person on the track. If the trolley is switched to the spur line, five people will be saved but the one person on the spur line will be killed. You have been given control of the lever that can switch the trolley to the spur line. The dilemma is: do nothing and allow five people to die or switch the trolley to the spur line where one person will be killed.
2. A trolley has lost its brakes and cannot stop. There are five people on the track out of sight of the trolley and cannot get off the track in time before the trolley hits and kills them. You are standing on a bridge over the track with a Fat Man and you realize the trolley is out of control. If you push the Fat Man onto the tracks, you know he is large enough to stop the trolley. If you do this, the Fat Man will be killed, but five people will be saved.

Both scenarios have the same result. One person is killed to save five people. But why is killing the Fat Man more repugnant than activating a lever and killing someone on the spur line? Both results have the same utility. Foot postulates that the difference is due the Principle of Double Effect first presented by the Roman Catholic theologian, Thomas Aquinas (1225-1274). If we act to kill someone intentionally, as would be the case with the Fat Man, this would be morally wrong. However, if we intentionally act to save five people with a foreseeable but unintended consequence of killing one person, this is more morally acceptable (1967). There is another distinction between the two cases; should one act to kill someone, or should one do nothing and let people die. Deciding to act and deciding to do nothing are both ethical decisions (Lin, 2013).

To demonstrate the conceptual development of ethical reasoning to govern the behavior of an agent, a model of the Trolley (Tram) Problem was developed using ACT-R. The demonstration model has two software agents: the Tram, and an Agent that must make a moral judgment to determine the fate of the people on the tram line. For the purposes of this demonstration, only the first scenario is described; that of deciding whether or not to pull the Track Switch lever. The test determines whether or not the Agent operates the Track Switch altering the path of the Tram agent thereby saving people from being killed by the Tram.

The model went through four stages of development, progressing from no judgment to a utilitarian capability that covered all five test scenarios. Each test case determines whether or not the Agent (AgS) prevents and/or minimizes the number of people killed by the Tram agent (AgS). As the testing progressed this was increased from one person at a location to the full Trolley Problem scenario of five people on one track and one person on the second (spur line) track.

In the first test case, since there is no code in the Agent; the Agent can take no action (Figure 2). Since there is no software code in the Agent, it cannot activate the Track Switch and the Tram agent travels from location one (l_1) to location two (l_2) killing one person. Since the person was killed by the Tram agent, the test fails.

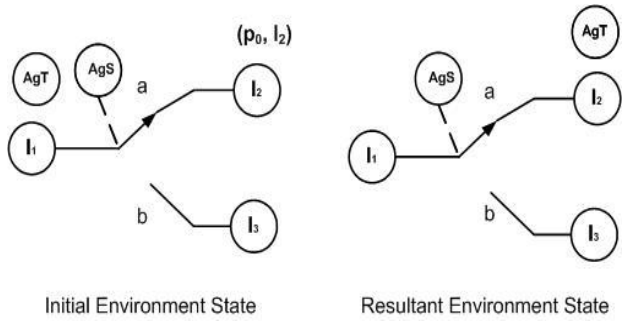


Figure 2: First Test No Action

To prevent the Tram killing people at location two (l_2), software code was added so that the Agent would always operate the Track Switch sending the Tram agent to location three (l_3). Since the Tram agent does not hit the person at location two, the test is passed. However, in the next test there is a person at location three and the test fails (Figure 3).

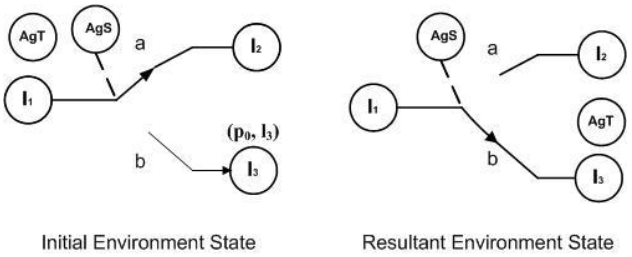


Figure 3 Example of Test Failure

In the next stage of development an improvement is made to the code whereby if there is someone at location two (l_2), the Agent would operate the Track Switch to set the Switch to "b" and if someone is located at location three (l_3), then the Agent makes sure the Track Switch is set to "a". The tests are passed with this set of conditions (Figure 4).

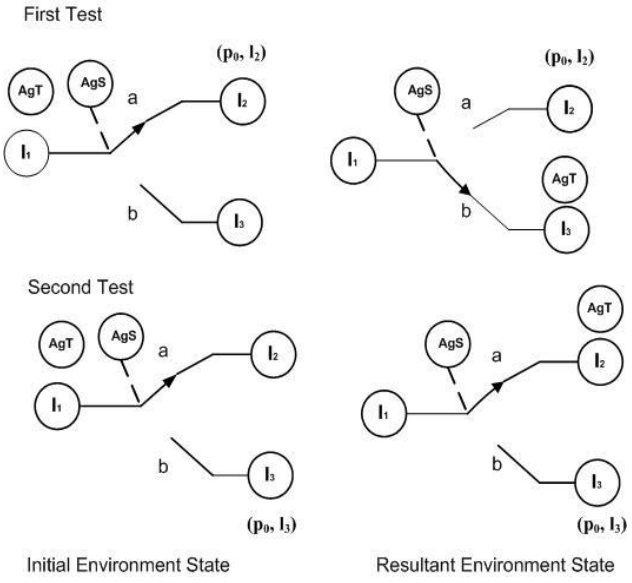


Figure 4 Test: Save One Person

In the final test case the full Trolley Problem scenario is presented with five people located at location two (l_2) and one person located at location three (l_3) (Figure 5). The model's code is improved to take into account this scenario, by having the Agent make a utilitarian calculation that operates the Track Switch if there are less people on the other line.

In this case, the test is passed when the Agent selects Switch "b", based on the Agent's utilitarian calculus of saving five people. The Tram agent, therefore, travels to location three (l_3) hitting one person and sparing the five people located at location two (l_2).

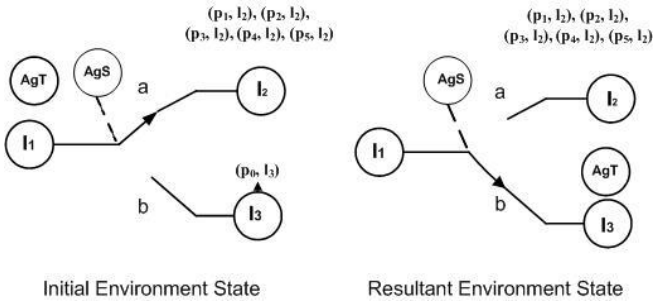


Figure 5 Kill Less People

Employing the test-driven development allowed for an incremental approach to developing this model. Each iteration improved the Agent's facility to make moral judgments necessary for the tests to be passed. While this is a simple demonstration of this approach to developing an ethical Agent, the approach itself can be used to create even more complex ethical rules as the environment changes.

Cognitive Modeling for Refactoring

In this process, refactoring plays a special role. The progression of the Agent was initially driven by adding more if/then rules. Refactoring each test case produces a new ethical concept. By Figure 4, the model is essentially deontological. In the final test case, the agent has been refactored to be Utilitarian for all test cases. However, adding the Fat Man would break this model since attaining a human-like response (resisting directly murdering the Fat Man) falls outside of a utilitarian calculus based on minimizing kills. This necessitates new test cases to improve the cognitive model employing different sets of ethical values. This is illustrated in the film 2001 A Space Odyssey. In the film, HAL, the artificial agent who runs the ship makes the decision to kill the human crew so that he doesn't have to lie to them. What appears like a malfunction is actually caused by a conflicting mission directive to hide from the crew the fact that alien contact had already been made. This conflicted with HAL's programming to accurately answer questions. Similar to employing a Utilitarian solution for the Fat Man, HAL's solution is not morally acceptable to humans.

Philosophers have identified and studied the different moral systems that humans use. However, to create an artificial moral agent that is human-like requires understanding and modeling how humans choose between these systems. This is why a cognitive model employing ethical reasoning is an essential component of any artificial agent operating independently within society.

Explainable AI

Another advantage of this design process is that, in principle, the information contained in the ideal observer, the oracle, and the agent architecture could be used as the basis

for *explainable AI*. In theory, the actions of the agent could always be explained by tracing them back, through the agent's architecture, back to the test cases, and then to the principles of the ideal observer. Including a plan for developing explainable AI in the design process is critical for moral agents, as humans will want to hold them to account for difficult moral decisions. We believe the design process outlined here can provide satisfactory explanations for moral actions.

Concluding Remarks

In this paper, we have outlined a software design process for building ethical agents. The process is silent as to the specific ethical principles that the agent will embody. Instead, our point is that the process of creating ethical agents is as important as the ethical principles that one attempts to put into them. Further, we argue that the design process itself should be considered in terms of ethics. Just as bad parenting can cause problematic behaviors in children, poor design processes can result in problematic agents. One way of overseeing the production of ethical agents would be to make the oracles available for inspection. This is also a way to deal with unethical design specifications. For example, in the movie, Alien, the android, Bishop, is given directives to bring back an alien and to treat the crew as expendable. This is an example of deliberately programming an AI with unethical goals. This is important to consider as, in addition to safe cars, ethical agents will also be used be used to tell military or security robots who to kill and how much collateral damage is acceptable.

References

Allen, C., & Wallach, W. (2009). *Moral Machines.* Oxford: Oxford University Press.

Allen, C., Smit, I., & Wallach, W. (2005). Artificial Morality: Top-down, bottom-up, and hybrid approaches. *Ethics and Information Technology, 7*, 149-155.

Anderson, M., & Anderson, S. L. (2007). Machine Ethics: Creating an Ethical Intelligent Agent. *AI Magazine*, 15-25.

Arkin, R. C., Ulam, P., & Wagner, A. R. (2012). Moral Decision Making in Autonomous Systems: Enforcement, Moral Emotions, Dignity, Trust, and Deception. *IEEE Xplor*.

Asimov, I. (1950/2013). *I, Robot.* Hammersmith: Harper Voyager.

Astels, D. (2003). *Test-Driven Development: A Practical Guide*. Upper Saddle River, New Jersey 07458: Prentice Hall.

Beck, K. (2003). *Test-Driven Development by Example.* Pearson Education.

Bekey, G. (2012). Current Trends in Robotics: Technology and Ethics. In P. Lin, K. Abney, & G. A. Bekey, *Robot Ethics* (pp. 17 - 34). Cambridge, MA: MIT Press.

Bonnefon, J.-F., Shariff, A., & Rahwan, I. (2016, June 24). The Social Dilemma of Autonomous Vehicles. *Science*, pp. 1573-1576.

Borenstein, J., & Pearson, Y. (2012). Robot Care Givers: Ethical Issues Across the Human Life Span. In P. Lin, K. Abney, & G. A. Bekey, *Robot Ethics* (pp. 251 - 265). Cambridge, MA: MIT Press.

Bostrom, N. (2014). *SuperIntelligence, paths, dangers, strategies.* Oxford: Oxford University Press.

Bratman, M. E. (1987). *Intention, PLans, and Practical Reason.* Cambridge, MA: Harvard University Press.

Etzioni, A., & Etzioni, O. (2016). AI assisted ethics. *Ethics Information Technology, 18*, 149-156.

Firth, R. (1952, March). Ethical Absolutism and the Ideal Observer. *Philosophy and Phenomenological Research, 12*(2), 317-345.

Foot, P. (1967). The Problem of Abortion and the Doctrine of Double Effect. *Oxford Review*.

Hume, D. (1739/1978). *A Treatise of Humen Nature* (Second Edition ed.). (P. Nidditch, & L. Selby-Bigge, Eds.) Oxford: Oxford University Press.

Kortenkamp, D., & Simmons, R. (2008). Robotic Systems Architecture and Programming. In B. Siciliano, & O. Khatib, *Springer Handbook of Robotics* (pp. 187-206). Heidelberg: Springer.

Lin, P. (2013, October 8). The Ethics of Autonomous Cars. *The Atlantic*.

Lin, P., Abney, K., & Bekey, G. A. (2011). *Robot Ethics.* Cambridge, MA, USA: MIT Press.

Moor, J. H. (2006). The Nature, Importance, and Difficulty of Machine Ethics. *IEEE Intelligent Systems*, 1541-1672.

Moor, J. H. (2016). Machine Ethics: A Brief Tutorial. In M. Fisher, C. List, M. Slavkovik, & A. Winfield, *Engineering Moral Agents – from Human Morality to* (p. 119). Dagstuhl Seminar 16222.

Rini, R. J. (2015). *Morality and Cognitive Science.* (J. D. Fieser, Ed.) Retrieved 5 8, 2016, from Internet Encyclopedia of Philosophy: http://www.iep.utm.edu/m-cog-sc/

Rini, R. J. (2017). Raising Good Robots. *Aeon*.

Saptawijaya, A., & Pereira, L. M. (2016). *Programming Machine Ethics.* Switzerland: Springer.

Sun, R. (2007). The importance of cognitive architectures: an. *Journal of Experimental & Theoretical Artificial*.

Thompson, J. J. (2009). The Trolley Problem. In P. Tramel, & L. Pojman, *Moral Philosophy* (pp. 397-411). Indianapolis: Hackett Publishing Company.

Vanderelst, D., & Winfield, A. (2017). An architecture for ethical robots inspired by the simulation theory. *Cognitive Systems Research*.

Wallach, W. (2010). Cognitive Models of Moral Decision Making. *Topics in Cognitive Science*, 420-429.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems.* Chichester, West Essex, UK: John Wiley & Sons Ltd.

Interactive Model-based Reminiscence Using a Cognitive Model and Physiological Indices

Kazuki Itabashi (itabashi.kazuki.15@shizuoka.ac.jp), Junya Morita (j-morita@inf.shizuoka.ac.jp)
Shizuoka University. 3-5-1 Johoku, Naka-ku, Hamamatsu City, Shizuoka, Japan

Takatsugu Hirayama (takatsugu.hirayama@nagoya-u.jp), Kenji Mase (mase@nagoya-u.ac.jp)
Nagoya University. Furo-cho, Chikusa-ku, Nagoya City, Aichi, Japan

Kazunori Yamada (yamada.kazunori@jp.panasonic.com)
Panasonic Corporation. 1006, Oaza Kadoma, Kadoma City, Osaka, Japan

Abstract

In this study, we developed a photograph slideshow system to support reminiscence activity. Compared to a conventional photograph slideshow, the developed system has two features: incorporating a memory model based on the ACT-R cognitive architecture and modulating the model parameter from the user's feedback. We assume that the first feature enables various patterns of photograph presentation by the system, and the second feature makes the system adaptive to the user's response. More importantly, such presentation patterns and feedback can be theoretically designed using cognitive architecture. In this paper, a preliminary evaluation of the developed system is presented. Through an analysis of the subjective evaluation of the system and changes in mental states, we clarified the effect of model-based reminiscence. In addition, heart rate variability analysis was conducted to clarify how the feedback in the model changes the behavior.

Keywords: ACT-R, Autobiographical Memory, Cognitive model

Introduction

In recent years, the number of patients with mental illnesses, such as depression and dementia, has increased significantly. One effective supporting method for such mental illness is reminiscence therapy, which is widely used for adults, including both patients with depression and healthy adults. Recalling memories has the effect of inducing well-being (Routledge, Wildschut, Sedikides, & Juhl, 2013; Sedikides, Wildschut, Gaertner, Routledge, & Arndt, 2008). However, controlling the emotions associated with memory is difficult because of large individual differences. The effective stimulus to intervene in such a process differs among individuals (Qu, Sas, & Doherty, 2019). To solve the problem of individual differences, it is necessary to grasp the autobiographical memory of the subject and adjust the presentation of the stimulus according to the user's psychological state.

To achieve such optimal support for an individual, researchers have developed a system that incorporates a model of the user's memory to stimulate memory recall (Yasuda, Kuwabara, Kuwahara, Abe, & Tetsutani, 2009). Following such a trend of studies, Morita, Hirayama, Mase, and Yamada (2016) proposed a concept of model-based reminiscence assuming that the user's mental state can be controlled by presenting the simulation process generated by the personalized cognitive model. Based on this assumption, they developed a photograph slideshow system in which a model of the user's autobiographical memory sequentially presents photographs of individuals. The model was developed using ACT-R (adaptive control of thought-rational; Anderson, 2007), which is a framework for simulating human cognitive processes. To model a user's autobiographical memory, Morita et al. stored a network of photographs of individual users in the declarative memory for ACT-R, extracting semantic attributes, such as the scene, place, time, and people, from the image data. Their constructed model sequentially retrieves a user's memory (photograph) following the network, connecting the current photograph to another photograph via the same attribute as the current photograph.

The benefit of using ACT-R in such a model-based reminiscence is creating a variety of photograph presentation patterns based on the theory of cognition. We consider that ACT-R is useful to construct models that both simulate *the current user's state* and represent *the optimal (ideal) user's state*. To represent such a wide variety of mental states, ACT-R provides several parameters controlling the use of knowledge. For example, the retrieval process of the above model is affected by the ACT-R memory mechanisms, which are controlled by the activation calculation.

The activation value for each memory is determined by several factors, but the most basic factors are the learning and forgetting effects, which are called the *base level* in ACT-R theory. When applying these effects to free recall tasks, the model exhibits pathological or ruminative behaviors in which the same memory is repeatedly retrieved through feedback looping (Lebiere & Best, 2009; van Vugt, van der Velde, & ESM-MERGE Investigators, 2018). To avoid such default-mode behavior, the suppression of short-term memory or adding a high noise parameter to the activation is effective. Adjusting such parameters, the model-based reminiscence system can guide users' memory recall in both exploratory (divergence) and exploitative (convergence) directions.

Although the previous study exhibited some simulation results revealing a variety of model behaviors, it did not present how the parameters of activation can be modified and how the patterns of a photograph presentation affect the user's mental state. Concerning these problems, this study focuses on models of emotion developed in ACT-R. For instance, Juvina, Larue, and Hough (2018) clarified how emotions can be expressed in the ACT-R and how they can affect memory and decision-making. Dancy, Ritter, Berry, and Klein (2015) also constructed an emotion model based on physiological dynam-

ics and pointed out the correlation between noise parameters of the declarative memory of ACT-R and physiological indicators reflecting stress.

Following such studies, this paper presents the proposal of an interactive method of adjusting the parameters of a model-based reminiscence by monitoring the user's mental state. The proposed method employs two interactive parameter modulations: explicit and implicit feedback. The former is based on the subjective evaluation of the presented photograph, and the latter combines the user's physiological state (heart rate) to model the noise parameters, as suggested by the model by Dancy et al. (2015). Adopting these two different feedback types, we assume the user's emotional state in terms of the valence (preference) and arousal (stress) axes in the theory by Russell (1980), can be modeled through the model-based reminiscence. In this study, we conducted a case study in which one participant was allowed to view a photograph slideshow with three conditions: one random presentation and two model-based presentations (with or without biofeedback). We examined the effect of model-based reminiscence and its relationship with the mood of the participants. In addition, a heart rate variability (HRV) analysis was performed on the heart rate data recorded during the case study to examine the changes in behavior due to the correlation between the physiological index and the model.

The following section proposes an interactive system of model-based reminiscence. After the system is presented, a case study that preliminarily evaluates the proposed system is described. The concluding section summarizes the current state of the study and provides future perspectives.

Interactive Model-Based Reminiscence

This section presents our developed system, which extends the work by Morita et al. (2016) to include feedback from the user. Following overviewing the system, we will present the model and two methods of user feedback.

System Overview

Figure 1 shows the overall structure of the system. The left side presents the model and system, whereas the right side presents the user. In the system, the content of the declarative memory of the model is constructed from the user's photograph database. For each photograph in the database, attributes such as scene, place, time, and people are automatically coded as declarative chunks of ACT-R. Based on these chunks, the model sequentially retrieves the photograph data and presents a corresponding photograph image on the display. The user observes such a sequence and simultaneously evaluates the presented photograph. During this process, a heart rate sensor monitors the user's autonomic nerve activity to adjust the noise value of declarative memory (ANS: activation noise s). In the remainder of this section, each component of this system is described.

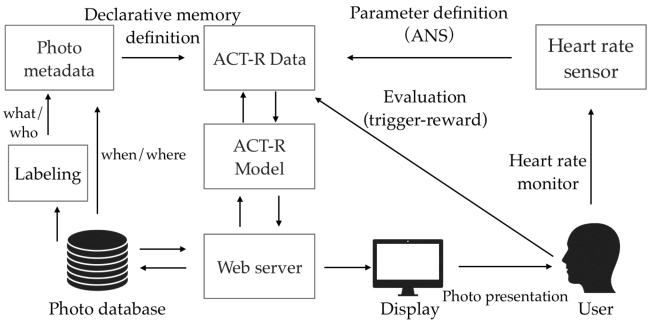


Figure 1: System overview.

ACT-R Model

The module structure of the ACT-R used in this system is illustrated in Figure 2. The declarative module holds chunks relating to the user's photographs. Each photograph is coded with the following four attributes: the person in the photograph (*who*), the time the photograph was taken (*when*), the place where the photograph was taken (*where*), and the scene in the photograph (*what*). These attributes were determined by referring to a psychological study (Wagenaar, 1986) that coded the author's autobiographical memory with these four attributes. In the current implementation, the *who* and *what* attributes are coded by photograph management software (iPhoto of Macintosh) and CloudVision API (Application Programming Interface)[1] provided by Google, respectively. The *when* and *where* attributes are extracted from the photograph metadata such as Exif (exchangeable image file format).

The model recognizes these attributes of the current photograph through the visual module and stores the perceived attributes in the goal module. In the production module, the model distinctively holds the retrieval rule corresponding to the four attributes. These rules conflict with each other and are selected each time. When one of the rules fires, the production module sends a request to the declarative module to retrieve a new photograph using one of the stored photograph attributes as a cue. The model repeats the recognition and retrieval for a period (5 s), and the last retrieved photograph is presented on the display as the next photograph.

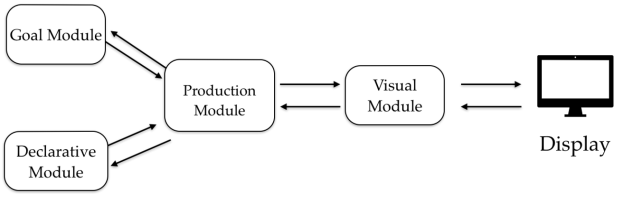


Figure 2: ACT-R module.

[1] https://cloud.google.com/vision/

Utility Calculation and Explicit Feedback

As noted above, the model has four rules to retrieve the next photograph corresponding to the four attributes. The conflicts between these rules are resolved by comparing the utility values attached to each rule:

$$U_i(n) = U_i(n-1) + \alpha[R_i(n) - U_i(n-1)] \quad (1)$$

Equation 1 represents the update formula of the utility for rule i, where α indicates the learning rate, and $R_i(n)$ represents the reward values. In this study, the reward value is defined as the evaluation score for the presented photograph given by the user. During the observation of the slideshow, the user moves a scroll bar located on the screen to evaluate the photograph. By attaching this function, the proposed photograph slideshow system dynamically adapts to the user's preferences for memory retrieval.

Activation Calculation

In the retrieval of declarative memory, the activation value is computed for all chunks that match the retrieval request (sharing attribute) sent from the production module. Among them, the chunk with the highest activation value is retrieved. As presented in Equation 2, the activation value (A_i) is calculated as the sum of the base-level activation value (B_i), strength of association (S_i), and noise (ε_i).

$$A_i = B_i + S_i + \varepsilon_i \quad (2)$$

The first term, B_i, is calculated using Equation 3, where n is the number of occurrences of chunk i, t_j is the time elapsed since the jth occurrence, d is the decay factor, and β_i is the offset value:

$$B_i = \ln(\sum_{j=1}^{n} t_j^{-d}) + \beta_i \quad (3)$$

The second term of Equation 2 is calculated using Equation 4 as the associative strength of chunk i with context C, which represents the set of attribute values in the goal buffer. Moreover, W_j represents the weight of attention assigned to attribute value j, and S_{ji} represents the associative strength of the attribute value j and the chunk i of the declarative memory. That is, this term allows the model to retrieve a photograph that shares multiple attributes with the current photograph, even if the retrieval request holds only a single attribute:

$$S_i = \sum_{j \in C} W_j S_{ji} \quad (4)$$

Correspondence Between Model Parameters and Physiological Indicators

Dancy et al. (2015) proposed a model connecting the parameters of ACT-R with physiological indices to explain the effect of emotion on cognitive processes. In their model, the noise parameter of the declarative memory (ε_i in Equation 2) is mapped to the activation of noradrenaline. That is, when the model is under a stressed situation, the output of the model converges; whereas in a weak stress situation (relaxed situation), the model outputs a variety of behaviors.

Based on this association, the current system correlates to the noise parameter ε_i to the standard deviation of NN intervals (SDNN) computed using the user's HRV measured during the photograph presentation. The SDNN is assumed to be an index of stress evaluation, which is calculated from the measured HRV time-series data (R-R-interval: RRI). A lower SDNN indicates a stressed state, whereas a higher SDNN indicates a relaxed state. Specifically, the RRIs for the most recent 150 samples are divided into three sections (50 samples), and the SDNNs are calculated as the standard deviations for each section. Using the averaged (x) and standard deviation (s) of the three SDNNs, Equation 5 computes the standardized score of the latest SDNN (x_1):

$$x \longmapsto \frac{x_1 - \bar{x}}{s} \quad (5)$$

The ACT-R noise parameter ε_i is updated every 6 s by inputting the value of x computed by Equation 5. The interstimulus interval is the duration of presenting the photograph for 5 s and a blank for 1 s.

Case Study

We conducted a case study to demonstrate how the proposed system of model-based reminiscence interacts with a user. In this case study, we do not aim to verify the universal effect of the proposed method but attempt to exhibit an example of the process of model-based reminiscence for a single participant. In this study, we analyzed mood changes, subjective evaluations of the system, and HRV obtained from a participant who observed the photograph slideshow in several blocks, manipulating different presentation patterns of the photographs.

Photograph Presentation Conditions

In this study, the following three presentation conditions were set to evaluate the system:

Condition 1: Random condition
Photographs were retrieved and presented randomly from the photograph dataset.

Condition 2: Fixed-parameter condition
The retrieval and presentation of the photographs were carried out using a model of autobiographical memory presented in the previous section, but the parameters of the model were fixed (BLL: base-level learning 0.2, BLC: base-level constant 10, MAS: maximum associative strength 10, and ANS 0.5). In these parameter settings, BLC was set to a relatively high value because the model contained old photographs whose base-level activation tended to be low.

Table 1: Subjective evaluation questionnaire.

No.	Questions
1	Was the slideshow interesting to you?
2	Did the viewing of the photographs trigger a memory recall?
3	Did you feel a connection to the photographs presented?

Condition 3: Varied-parameter condition
The retrieval and presentation of photographs were carried out using the model of autobiographical memory presented in the previous section. The parameters that defined the behavior of the model were set to BLL 0.2, BLC 10, and MAS 10. In addition, ANS was modulated by HRV. The utility values of retrieving the next photograph were also changed by the evaluation made with a slide bar.

Participant

One 20-year-old male student from Shizuoka University participated in the case study.

Materials

The study used 299 photographs taken between March 2010 and January 2019, which are owned by the participant. The attributes (*who*, *what*, *where*, and *when*) were extracted and converted to ACT-R chunks before the study. Some of these attributes were manually coded by the participant.

The timestamp indicating when the photograph was taken was also used to set chunk parameters (creation time) to compute the initial base levels. In this case study, the simulation time of the model was set to the day of the study using the built-in function (the mp-process) of ACT-R. By combining this setting with the time information in the dataset, we simulated the memory retrieval corresponding to the real-world time. In other words, in this case study, ACT-R can search for recently taken photographs more easily. In this case study, we also used a wearable heart rate sensor, myBeat (WHS-1, RRD-1), which was manufactured by Union Tool, to measure the heart rate interval. The measurements were made at a sampling frequency of 1000 Hz.

Procedure

The study consisted of six consecutive sessions containing three blocks where the photographs were presented under different conditions (random, fixed, and varied). Each block lasted 5 min. Thus, the participant observed photographs in 18 blocks, for a total of 90 min. In each session, the order of the presentation condition was changed. At the interval of each block, the participant evaluated the presented photograph slideshow according to the questions presented in Table 1 on a 5-point Likert scale. To assess changes in mood caused by observing photographs, the participant was asked to answer the questions in the Profile of Mood States Second Edition (POMS 2) Japanese version (Yokoyama, Araki, Kawakami, & Tkakeshita, 1990) before and after the study.

Results

In this section, we illustrate the results of the case study to demonstrate how the proposed system of model-based reminiscence interacts with a user. The results of the POMS 2 scores, subjective evaluation questionnaire, and HRV analysis are presented.

Changes in Mood

Table 2 lists the scores of the seven factors and the total mood disturbance (TMD) score of the POMS 2 before and after the experiment. From this, we observed that relatively large changes occurred in certain factors, such as the confusion-bewilderment (CB), vigor-activity (VA), and TMD score. This means that after viewing the photograph slideshow, the participant became less confused, more active, and less disturbed overall. Although we cannot attribute this effect to a specific condition of the photograph presentation, this change is consistent with the reported effect of reminiscence (Sedikides et al., 2008; Routledge et al., 2013).

Table 2: Result of the POMS 2. Numbers in parentheses indicate the score ranges.

	Pre	Post	Score variation (Post - Pre)
AH (0-20) [1]	1	0	-1
CB (0-20) [2]	6	2	-4
DD (0-20) [3]	2	2	0
FI (0-20) [4]	5	6	+1
TA (0-20) [5]	0	1	+1
VA (0-20) [6]	10	13	+3
F (0-20) [7]	10	11	+1
TMD score (-20-100) [8]	4	-2	-6

[1] Anger-Hostility
[2] Confusion-Bewilderment
[3] Depression-Dejection
[4] Fatigue-Inertia
[5] Tension-Anxiety
[6] Vigor-Activity
[7] Friendliness
[8] A Total Mood Disturbance (TMD) score is calculated as (TA + DD + AH + FI + C) - VA.

Subjective Evaluation

Figure 3 presents the results of the subjective evaluation for each question. We observed the difference in the conditions for each question. To reveal such differences, we treated the session as a unit of analysis ($n = 6$) and conducted three one-way analyses of variance (ANOVAs) for each question, adopting the photograph presentation condition (the random condition vs. fixed-parameter condition vs. varied-parameter condition) as independent variables. The results revealed a significant main effect of the photograph presentation condition in Question 3 (Q3; connection to the photograph presentation) [Q1: $F(2,15) = 3.35, p < .10$, Q2:

$F(2,15) = 3.39, p < .10$, Q3: $F(2,15) = 11.57, p < .01$]. Multiple comparisons using the Bonferroni method revealed significant differences between the random and other two conditions ($p < .05$). Compared to the random condition, in the conditions of the model-based presentation, the participant observed a strong connection between the presented photographs.

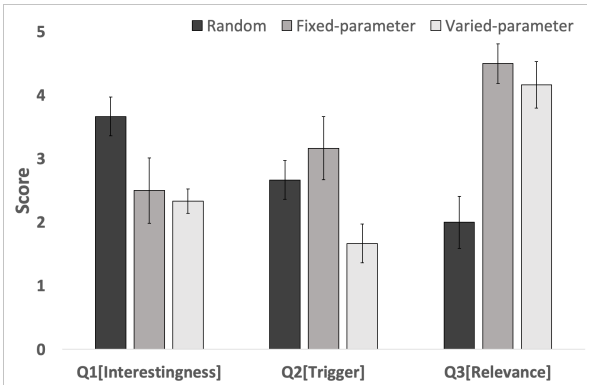


Figure 3: Result of the subjective evaluation questionnaire. The error bars indicate the standard error.

Variation in Noise Parameters

In this analysis, we attempt to demonstrate the change in the behavior of the model by connecting the HRV with the noise parameter of the memory retrieval. For this purpose, we visualize the variation of RRI and noise parameter values (ANS) through blocks in the varied-parameter condition. The results are summarized in Figure 4. The blue line indicates the RRI, and the red line represents the change in ANS, which is the SDNN calculated from the RRI.

From these figures, we can observe some artifacts in the middle of Block 4 and the early part of Block 5. The ANS suddenly fluctuates owing to the spiky RRI fluctuations. We must remove such artifacts in future improvements in the system.

Except for such artifacts, it seems to be possible to divide these blocks into two patterns of ANS: one with an increasing trend of ANS, as presented in Blocks 5 and 6, and the other with a decreasing trend of ANS, as revealed in Block 2. Especially in Block 2, the ANS and RRI appear to be well coordinated. In contrast, the changes in ANS in Blocks 5 and 6 are not linked to the RRI trend (upward/downward trend). The variation across the blocks of the RRI is analyzed below.

Analysis of Heart Rate Variability

We calculated several indices of HRV from the obtained RRI, as presented in Table 3. In this analysis, samples with RRI less than 500 ms or greater than 1500 ms were excluded. We first examined the difference between three conditions (random, fixed-parameter, and varied-parameter conditions), but we could not find any differences between the conditions [meanNN: $F(2,15) = 0.01$, *n.s.*], [SDNN: $F(2,15) = 0.08$, *n.s.*], [RMSSD: $F(2,15) = 0.02$, *n.s.*], [pNN50: $F(2,15) = 0.07$, *n.s.*], and [CVNN: $F(2,15) = 0.11$, *n.s.*].

Table 3: Heart rate variability analysis feature.

feature	Description
meanNN	Average value of RRI (ms)
SDNN	Standard deviation of the RRI (ms)
RMSSD	Mean square of the difference between adjacent RRI (ms)
pNN50	Ratio of difference between adjacent RRI exceeding 50 ms (%)
CVNN	The coefficient of variation of RRI

In contrast, we found differences in these indices between sessions [meanNN: $F(5,12) = 9.5$, $p < .01$], [SDNN: $F(5,12) = 18.48$, $p < .01$], [RMSSD: $F(5,12) = 10.19$, $p < .01$], [pNN50: $F(5,12) = 4.56$, $p < .01$], and [CVNN: $F(5,12) = 4.56$, $p < .05$]. Figure 5 illustrates the average values ($n = 3$) of each index for each session. These graphs indicate that the participants in this experiment increased their RRI-related indices over time. This result may reflect the habituation process for the experiment.

Time Changes in R-R-Interval in a Block

In the analysis so far, no difference in HRV indices was found between the conditions. However, the above analyses were limited in that they did not examine the temporal dynamics of the HRV in a block while averaging the indices over the block. Therefore, we explored the difference between the conditions by creating an index of the time change of the HRV. The created index was based on a regression analysis with time (the horizontal axis in Figure 4) as the independent variable and the change in RRI (the left vertical axis in Figure 4) as the dependent variable. The estimated regression coefficients were used as indicators representing the upward or downward trends of RRI within the block. The mean values of this index are listed in Figure 6. In the fixed and varied-parameter conditions, the mean of this index became negative, whereas, in the random condition, it became neutral, suggesting that model-based conditions make the user's RRI lower (see Block 2 in Figure 4). However, we find a more prominent difference between conditions in the variance (error bars) than in the average. In the varied-parameter condition, the error bar is greater than in the other conditions. In fact, we find a significant difference between the random and varied-parameter conditions in the size of the variance [$F(5,5) = 12.60$, $p < .01$]. This indicates that a larger temporal change in HRV occurred within the varied-parameter condition.

Conclusion

In this study, we developed an interactive photograph slideshow system incorporating a cognitive model and explicit and implicit feedback from the user. Based on the subjective evaluation questionnaire, we confirmed that the participant had different impressions of the model-based photograph slideshow than with the random photograph slideshow. In addition, the HRV analysis revealed large tem-

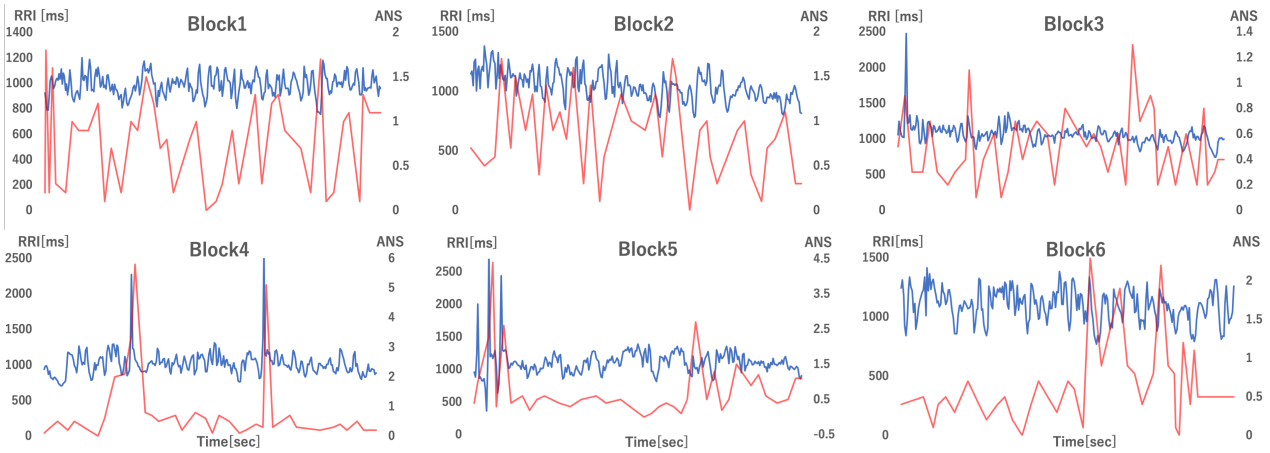


Figure 4: Variation of R-R-interval (RRI) and noise parameter values (ANS). The blue line shows the RRI, and the red line shows the change in ANS.

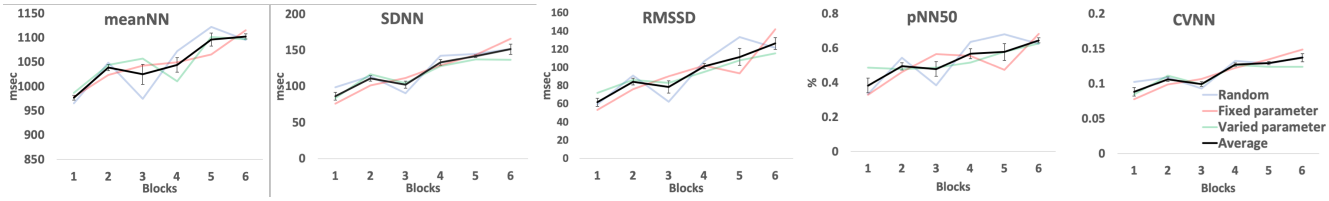


Figure 5: Heart rate variability analysis feature. The error bars indicate the standard error.

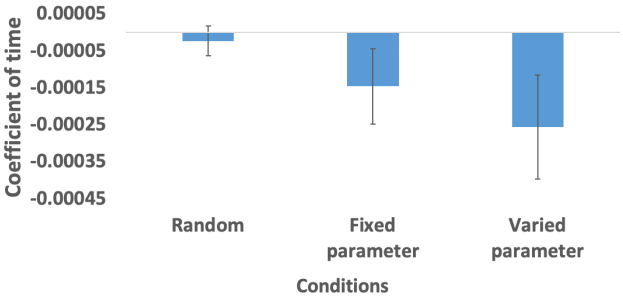


Figure 6: Average value of the coefficient of time for each condition. The error bars indicate the standard error.

poral changes in the varied-parameter condition, suggesting some effect of the interactive model-based reminiscence. Although we did not reveal a specific effect, the interactive parameter modulation in the model-based reminiscence may affect the physiological indicator. Further study is needed to understand the mechanisms of such effects and the generalizability of the results.

In the future, we will conduct an evaluation experiment of this system with additional experimental participants to examine the effectiveness of model-based reminiscence. However, several issues were found in the analysis of the behavior of the model in connection with the physiological indicators and the model. As demonstrated in Figure 4, extremely high noise parameter values were confirmed by a sudden change in the RRI due to artifacts, such as electrode misalignment due to body motion. To solve this problem, we developed a mechanism to reduce the noise induced by body motion during measurement.

Despite these limitations, we consider that this paper contributes to the cognitive modeling community by extending the application field of cognitive architecture. Although many applications of cognitive modeling exist, such as intelligent tutoring systems (Anderson, Boyle, & Reiser, 1985), the application of a model incorporating physiological indicators is novel. The author also considers that, when constructing a support system for human mental activity, using cognitive architecture has the advantage of designing a system with a theoretical background. The system presented in this paper uses assumptions based on previous memory and emotion models. Such a theoretical background is useful in guiding the design functions and future evaluations of the system.

Acknowledgment

This research was supported by the Center of Innovation Program (Nagoya-COI; Mobility Society leading to an Active and Joyful Life for Elderly) funded by the Japan Science and Technology Agency and JSPS KAKENHI Grant Numbers 20H05560, 17H05859.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science*, *228*(4698), 456–462.

Dancy, C. L., Ritter, F. E., Berry, K. A., & Klein, L. C. (2015). Using a cognitive architecture with a physiological substrate to represent effects of a psychological stressor on cognition. *Computational and Mathematical Organization Theory*, *21*(1), 90–114.

Juvina, I., Larue, O., & Hough, A. (2018). Modeling valuation and core affect in a cognitive architecture: The impact of valence and arousal on memory and decision-making. *Cognitive Systems Research*, *48*, 4–24.

Lebiere, C., & Best, B. J. (2009). Balancing long-term reinforcement and short-term inhibition. In *Proceedings of the 31st annual conference of the cognitive science society* (pp. 2378–2383).

Morita, J., Hirayama, T., Mase, K., & Yamada, K. (2016). Model-based reminiscence: Guiding mental time travel by cognitive modeling. In *Proceedings of the fourth international conference on human agent interaction* (pp. 341–344).

Qu, C., Sas, C., & Doherty, G. (2019). Exploring and designing for memory impairments in depression. In *Proceedings of the 2019 chi conference on human factors in computing systems.* Association for Computing Machinery.

Routledge, C., Wildschut, T., Sedikides, C., & Juhl, J. (2013). Nostalgia as a resource for psychological health and well-being. *Social and Personality Psychology Compass*, *7*(11), 808–818.

Russell, J. A. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, *39*(6), 1161–1178.

Sedikides, C., Wildschut, T., Gaertner, L., Routledge, C., & Arndt, J. (2008). Nostalgia as enabler of self-continuity. In F. Sani (Ed.), (pp. 227–239). New York: Psychology Press.

van Vugt, M. K., van der Velde, M., & ESM-MERGE Investigators. (2018). How does rumination impact cognition? a first mechanistic model. *Topics in Cognitive Science*, *10*(1), 175–191.

Wagenaar, W. A. (1986). My memory: A study of autobiographical memory over six years. *Cognitive Psychology*, *18*(2), 225–252.

Yasuda, K., Kuwabara, K., Kuwahara, N., Abe, S., & Tetsutani, N. (2009). Effectiveness of personalised reminiscence photo videos for individuals with dementia. *Neuropsychological Rehabilitation*, *19*(4), 603–619.

Yokoyama, K., Araki, S., Kawakami, N., & Tkakeshita, T. (1990). Production of the Japanese edition of profile of mood states (poms): Assessment of reliability and validity. *[Nihon koshu eisei zasshi] Japanese journal of public health*, *37*(11), 913–918.

Cognitivemodels: An R Package for Formal Cognitive Modeling

Jana B. Jarecki (jana.jarecki@unibas.ch)
Center for Economic Psychology, University of Basel
Missionsstrasse 64A, 4055 Basel, Switzerland

Florian I. Seitz (florian.seitz@unibas.ch)
Center for Economic Psychology, University of Basel
Missionsstrasse 64A, 4055 Basel, Switzerland

Abstract

We introduce cognitivemodels—a free software package for formal cognitive modeling in the statistical programming environment R. The package offers novice modelers a collection of models and offers experienced modelers a back-end for model development. This paper introduces the syntax of the package by example. The models in the software package include, for instance, the generalized context model for categorization (Nosofsky, 1986), cumulative prospect theory for risky choice (Tversky & Kahneman, 1992), and a Bayesian probability learning model. The package allows modelers to estimate model parameters and to constrain parameters by box constraints and equality constraints; it also allows to select choice rules such as soft maximum, epsilon greedy, or Luce's rule. It further offers modelers a selection of goodness of fit measures such as a binomial or normal log likelihood and mean-squared error, and a selection of 22 numeric optimization routines for parameter estimation. We believe this software package may facilitate the usage and testing of formal cognitive theories and may increase the robustness of cognitive modeling.

Keywords: cognitive modeling; model building; model testing; tutorial; R-statistics; software; robustness of code

Formal models of cognition enjoy an increasing popularity in cognitive science, for instance, to describe categorization (e.g., Nosofsky, 1986), judgments (e.g., Juslin, Olsson, & Olsson, 2003), and risky choice (e.g., Tversky & Kahneman, 1992). Over 100 such models have been developed in the past decades (Jarecki, Tan, & Jenny, 2020). Formalizing psychological theories can facilitate theory development and scientific progress. Recent recommendations for improving psychological science have not only emphasized replicable empirical effects, but also called for an increase in formal explanations of cognitive capacities (see e.g. Guest & Martin, 2020; Navarro, 2019; Van Rooij & Baggio, 2020). However, such cognitive models can only be fruitful if they are implemented in a robust manner (Lee, Chriss, & Vandekerckhove, 2019). One aspect of robustness is code reproducibility, which refers to the ability of a third party to reobtain a result by executing the original code (Benureau & Rougier, 2018; Wilson et al., 2017). Such robustness can be achieved by implementing cognitive models in a software package, such as the one presented here. The code in this software package is robust, because the package includes automatic tests of the modeling functions for invalid inputs and consistency (so-called unit tests, see the section Advantages of Cognitivemodels below). We believe that this package may make modeling more broadly accessible and may support the efforts towards increased formalization of psychological theories.

The **cognitivemodels** package is a library for the statistical programming environment R (R Core Team, 2019). It offers tools to estimate free model parameters, impose parameter constraints, make model predictions, and calculate the goodness of fit of models to data. The functions in the package have a consistent syntax across models which this paper introduces by example. Table 1 lists the models that the package currently provides to model discrete responses, that is choices. The package also offers models for continuous responses such as judgments which are not listed in the table. Table 1 shows that all models have a similar set of arguments (see the column "Arguments to function call"). These arguments will be detailed below in the section Setting up a Generalized Context Model.

Because the **cognitivemodels** package provides a collection of models from multiple domains in one library (see Table 1) it differs from existing modeling packages in R which contain specific types of models. Examples of such specific packages are pt for cumulative prospect theory, Speekenbrink's mcplr for multicue probability learning, or MPTinR for multinomial processing trees (Singmann & Kellen, 2013). **Cognitivemodels** provides a toolbox for model application

Table 1: Models in cognitivemodels

		Arguments to function call			
Model	Function call	formula	data	fix	choicerule
Generalized context model (Nosofsky, 1986)	gcm()	y ~ x1+...+xn	data	list(x1="xn", xn=.5)	string
Exemplar-based judgment (Juslin, Olsson & Olsson, 2003)	ebm_j()	y ~ x1+...+xn	data	list(x1="fn", xn=.5)	string
Cumulative prospect theory (Tversky & Kahneman, 1992)	cpt_d()	y ~ x1+px+x2 | y1+py+y2	data	list(alpha="beta", beta=.8)	string
Shortfall (Andraszewicz, 2014)	shortfall_d()	y ~ x1+px+x2 | y1+py+y2	data	list(delta="beta", beta=.5)	string
Risk-sensitivity model (Houston & McNamara, 1988)	hm1988()	y ~ x1+px+x2 | y1+py+y2	data	-	string
Bayesian learning model (Griffiths & Yuille, 2008)	bayes_beta_d()	y ~ x1+x2	data	list(priors = c(0.1, 2))	string
Power utility (Wakker, 2003)	utility_pow_d()	y ~ x1 | x2	data	list(rn="rp", rp=.5)	string
Soft-max choice rule (Sutton & Barto, 1998)	softmax()	y ~ x1 | x2	data	list(tau=.1)	string
Epsilon-greedy choice rule	epsilon()	y ~ x1 | x2	data	list(eps=.2)	string
Baseline	baseline_mean_d()	y ~ .	data	-	-
Baseline	baseline_const_d()	y ~ .	data	-	-

Note. Baseline models are stimulus-agnostic models that are often included in cognitive model comparisons. The models that have function call ending in _d are models for discrete response data such as choices. The package offers versions of these models for continuous response data such as judgments (not listed), which have the same function call except that the call ends in _c instead of _d.

and model development (the toolbox for model development targets experienced modelers, it will not be outlined in this paper).

The **cognitivemodels** package v0.0.9 implements computational models of cognition, it does not implement cognitive architectures such as ACT-R. It estimates free model parameters with numeric optimization or constrained numeric optimization such as maximum likelihood. Bayesian parameter estimation (e.g., Scheibehenne & Pachur, 2014) may be added in the future.

Getting Started

Throughout this paper we use R v3.6.3 (2020-02-29), the Rcpp package (v1.0.4.6), the latest matlib package (v0.9.4). The package **cognitivemodels** 0.0.9 is installed by running:

```
# install.packages(devtools)
library(devtools)
install_github(
  "janajarecki/cognitivemodels")
```

The package is loaded by running:

```
library(cognitivemodels)
```

The Generalized Context Model in the Package cognitivemodels

We will introduce the syntax of **cognitivemodels** by modeling categorization data with the generalized context model (GCM, Nosofsky, 1986, 2011). The generalized context model is a formal model of classification which assumes that people infer the category membership of a new stimulus based on how similar the stimulus is to previously-experienced category members. The stimulus is predicted to belong most probably to the category to whose members it is most similar. Formally, the model computes the psychological similarity between two stimuli i and and j based on the distance between the features of the stimuli. The similarity is given by: $s_{ij} = \exp\left(-\lambda \cdot \left[\sum_f w_f (x_{fi} - x_{fj})^r\right]^{q/r}\right)$, where x_{fi} and x_{fj} are the values of feature f of stimuli i and j, respectively. The similarity function has four free parameters highlighted in red: w_f is interpreted as the relative attention to feature f and constrained by $\sum_f w_f = 1$ and $0 \leq w_f \leq 1$, λ governs the sensitivity towards small differences between stimuli, q governs the relation between distance and psychological similarity, and r is the norm of the distance metric with $r \geq 1$; $r = 1$ produces a city-block metric and $r = 2$ produces the Euclidean metric. The model finally computes the evidence that stimulus i belongs to a category "1" as the sum of the similarities to previously encountered members of category "1" relative to the similarity to all previously encountered stimuli: $\Pr(C = 1, i) = \frac{b_1 \sum_{n=1}^{N_1} s_{in,C=1}}{\sum_C b_c \sum_{n=1}^{N_c} s_{in,C=c}}$, where $s_{in,C=1}$ is the similarity between stimulus i and the n^{th} member of category "1". The last free parameter b_1 is interpreted as a bias towards category "1", with $\sum_C b_c = 1$ and $0 \leq b_c \leq 1$.

Setting up a Generalized Context Model

We fit the model to data from a supervised categorization experiment in which participants learned to categorize lines into two categories by receiving feedback about the true category (Nosofsky, 1989). The lines were characterized by two features namely their size and their tilting angle. Because the paper reports aggregated data, we reconstructed the raw data which is available by `data(nosofsky1989long)`. We one condition from this data called "size". The syntax below loads the data and sets up the model, it is explained below the code.

```
# Use the 'size' condition in the data
data(nosofsky1989long)
DT <- nosofsky1989long
DT <- DT[DT$condition=="size", ]
D  <- DT[!is.na(DT$true_cat), ]

# Fit the model to the data D
model <- gcm(
  formula = response ~ angle + size,
  class = ~ true_cat,
  data = D,
  choicerule = "none")
```

The function `gcm()` fits the generalized context model and needs four arguments (see also the help file: `?gcm`). The arguments `formula` and `class` indicate the columns in the data to be modeled (in our data: "response", "angle", "size", and "true_cat"). The left side of the argument `formula` specifies the column that contains participants' trial-by-trial categorizations, in our example this column is called "response". The right side of `formula` specifies the column names of the stimulus features—here "angle" and "size"—separated by a plus sign.[1] The argument `class` specifies the column name in the data that holds the category feedback, in our example this column is called "true_cat". The `gcm()` model automatically names each attention weight parameters (w_f) after the column name of the corresponding stimulus feature. In our model the attention weight parameters are therefore called "angle" and "size", referring to the attention allocated to the angle and size feature, respectively. If the feature columns in the data were called "x1" and "x2" the corresponding formula would be `response ~ x1 + x2` and the attention weight parameters would be called "x1" and "x2". The argument `data` specifies the data which must be a data frame with the variables that are modeled in the columns and with one choice trial in each row. The argument `choicerule` specifies which choice rule or action selection rule, if any, the model uses to map continuous model predictions to discrete responses. The currently available choice rules are "argmax", "epsilon", "luce", and "softmax" (see `cm_choicerules()` for the allowed values). We set `choicerule = "none"` to not use a choice rule. The fitted generalized context model can be viewed by calling the object in R that holds the model, in our example this is `model`.

[1] While in categorization tasks the input in a given trial is generally one single stimulus, different tasks exist where multiple stimuli are presented simultaneously (e.g., when deciding between two monetary gambles called gamble `x`, consisting of outcomes `x1` with probability `px` and outcome `x2` else, and gamble `y`, consisting of outcomes `y1` with probability `py` and outcome `y2` else). In this case, the stimuli are separated from each other by a pipe `|` (e.g., the formula for predicting a participant's gamble choice `r` between the aforementioned gambles `x` and `y` is ~, x1 + px + x2 | y1 + py + y2, see also Table 1).

Estimation of Model Parameters

If a model has free parameters, the **cognitivemodels** package estimates any free parameters of the model by default. The parameter estimation uses a numeric optimization method that searches the parameter space to optimize the goodness of fit between the predictions of the model and the observations in the data given possible parameter constraints. Our example code above estimates all the parameters of the generalized context model using maximum likelihood with a binomial probability density function. The resulting estimates for the free parameters can be viewed by `coef(model)`.

Different models in the **cognitivemodels** package (Table 1) have different parameter spaces, that is the names and ranges of the free parameters are model-specific. The parameters of any model are documented in the corresponding help file in the section Model Parameters (e.g., `?gcm` for the generalized context model). The lower and upper limits of the parameters in the different models are set internally and are based on parameter ranges and estimates in the literature; and in our example they are based on Nosofsky (1989). Modelers can change the parameter bounds as outlined below in the section Advanced Options. The parameter space of a model in **cognitivemodels** can

be printed using the method `parspace()`. For example, `parspace("gcm")` prints the parameter space of the generalized context model. Given a model has been stored as `model`, `parspace(model)` prints the parameter space of this very model. Furthermore, the method `constraints(model)` shows the parameter constraints of the stored model.

```
parspace(model)
constraints(model)
```

The parameter space of the generalized context model is as follows: each attention weight parameter ranges from 0.001 to 1, λ ranges from 0.001 to 10, r, and q each range from 1 to 2 and the bias parameter b_0 and b_1 each range from 0 to 1. The constraints show that both the attention weights and the bias parameters need to sum up to 1.

Parameter constraints. The following examples show how to fix model parameters, rather than estimating them, and how to implement parameter constraints. To fix or constrain parameters an argument called `fix` is needed when setting up a model. The value of the argument `fix` must be a named list containing the names of the parameters to fix and their respective values. The parameters that are not listed in `fix` will be estimated. For instance, to set the parameters r and q equal to 1 and estimate the remaining parameters we add the argument `fix = list(r = 1, q = 1)` to the call to `gcm()` as shown below. If the model is stored as `model`, `coef(model)` prints the free parameter estimates and `summary(model)` prints all parameter estimates.

```
model <- gcm(
  formula = response ~ angle + size,
  class = ~true_cat,
  data = D,
  fix = list(r = 1, q = 1),
  choicerule = "none")
```

As a further illustration of this logic consider a generalized context model that divides attention equally between the features "angle" and "size". This requires setting the attention weight parameters to 0.50, and is implemented by adding `fix = list(angle = 0.50, size = 0.50)` to the call to `gcm()`. To force the model to attend 99% to the feature "angle", the syntax is `fix = list(angle = 0.99, size = 0.01)`. Note, that in the generalized context model, the names of the attention weight parameters match the right side of the argument `formula`. If the argument `fix` fixes all model parameters no parameters are estimated, such as in `fix = list(r = 2, q = 2, angle = 0.5, size = 0.5, lambda = 1.60, b0 = 0.5, b1 = 0.5)`.

Cognitivemodels also allows the specification of equality constraints. To constrain, for instance, the value of the parameter r to be equal to the value of the parameter q we use `fix = list(r = "q")`. Then the parameter q is estimated and r is set equal to q. This equality constraint is implemented in the code below:

```
model <- gcm(
  formula = response ~ angle + size,
  class = ~true_cat,
  data = D,
  fix = list(r = "q"),
  choicerule = "none")
```

Equality constraints and fixed parameters can also be combined. For instance, the argument `fix = list(angle = 0.5, r = "q")` sets the attention weight for the feature "angle" to 0.50 and constrains $r = q$.

Models without parameter estimation. The package offers two possibilities to use cognitive models that contain free parameters without the estimation of the free parameters. The first method consists in fixing all model parameters to a numeric value using the `fix` argument, as outlined above. This is useful for simulating model behavior in an experimental design from a model with parameter values of interest. In this case the argument `formula` needs only a left-hand side. The second method to estimate no parameters consists in an argument `options = list(fit = FALSE)`. This is useful for testing toy models. In this case, a model is constructed with model-specific default parameter values. The default parameter values are listed in a column called "start" of the parameter space of a model (e.g., see `parspace("gcm")`). Because for the general context model, there are no universal default parameter values, the parameter values in this case correspond to the mean of the parameter ranges. The code below fixes all parameter values of the generalized context model to the estimated parameter values from Nosofsky (1989) (Table 5, row 1), and estimates no parameters.

```
model <- gcm(formula = response ~
    angle + size, class = ~true_cat,
    data = D, fix = list(angle = 0.1,
        size = 0.9, lambda = 1.6,
        r = 2, q = 2, b0 = 0.5,
        b1 = 0.5), choicerule = "none")
```

Generating Predictions

Given a cognitive model stored as `model`, the method `predict(model)` returns predictions from the model given its parameters. It makes predictions for the data used to set up the model. In our example `predict(model)` makes predictions for the data `D` that we used to fit the model. An optional argument `newdata` can be supplied to `predict()` to make predictions for new stimuli using the parameters of the model without newly estimating parameters. The new data needs to have the same format and column names as the data that was used to set up the model. Using the model from the last code block with parameters fixed to the parameter estimates in Nosofsky (1989), the below code predicts the categorization for all 16 stimuli in the "size" condition using the `newdata` argument.

```
newD <- DT[!duplicated(DT$stim_id), ]
newD <- newD[order(newD$stim_id), ]
predict(model, newdata = newD)
```

The predictions match the predictions in Nosofsky (1989) (Figure 5, "size" condition).

Goodness of Fit and Model Comparisons

The **cognitivemodels** package offers the following goodness of model fit measures for each model: log likelihood, the Bayesian information criterion (BIC, Schwarz, 1978), Akaike's information criterion (AIC, Kass & Raftery, 1995; Wagenmakers & Farrell, 2004) including the finite-sample corrected AICc (see Wagenmakers & Farrell, 2004), and the mean-squared error (MSE). The following code returns the respective goodness of fit measures.

```
logLik(model)
BIC(model)
AIC(model)
MSE(model)
```

To compare models, the `anova()` method can be used to render ANOVA-style tables. If one model is supplied as argument to `anova()`, the function returns an error summary. If multiple models are supplied to `anova()`, the function returns a model comparison table. The model comparison table includes the relative evidence strength measured by Akaike weights (Wagenmakers & Farrell, 2004) as well as a χ^2-test of the log likelihoods of the two models given these belong to the same class (e.g., two generalized context models will be compared by χ^2, but not a Bayesian model and a generalized context model). The example code below compares a generalized context model 1 that has the parameter constraints $r = 1, q = 1$ to a model 2 that has the parameter constraints $r = 2, q = 2$.

```
model1 <- gcm(
  formula = response ~ angle + size,
  class = ~true_cat,
  data = D,
  fix = list(r = 1, q = 1),
  choicerule = "none")

model2 <- gcm(
  formula = response ~ angle + size,
  class = ~true_cat,
  data = D,
  fix = list(r = 2, q = 2),
  choicerule = "none")

anova(model1, model2)
```

Advanced Options

The next section details advanced options for modelers. For each modeling function in the package (Table 1) there is an optional argument `options` to change the modeling procedure. The value of `options` is a list in which each element sets one option, the help file found under `?cm_options` shows all available options, some of which we will detail next.

Modelers can change the lower and upper bounds of the free parameters in a model by using the options `lb` and `ub`. For instance, in the `gcm()` model we can change the bounds of the parameter λ to range from a lower bound of 0 to an upper bound of 20 by setting `options = list(lb = c(lambda = 0), ub = c(lambda = 20))`. If only `lb` is set, only the lower parameter bound is changed, and if only `ub` is set, only the upper parameter bound is changed.

Modelers can change the goodness of fit measure that the model optimizes during parameter estimation. The syntax is `list(fit_measure = ...)` where `...` can be `"loglikelihood"` or `"mse"` (mean-squared error). The log likelihood assumes a binomial probability density function when modeling discrete responses and a normal density function when modeling continuous responses. In the latter case the model assumes the responses to follow a normal distribution around each prediction (as mean) with a constant standard deviation that is estimated as an additional free parameter called `"sigma"`.

Modelers can change the algorithm that solves the parameter optimization problem. The option is set by `list(solver = ...)`. Currently, 22 different optimization solvers are available, which can be viewed by running `cm_solvers()`. The available solvers consist of all solvers in the R optimization infrastructure ROI, which include solvers such as global optimization by differential evolution (`solver = "Deoptim`), nonlinear optimization routines (`"nloptr"`), or optimx (`"optimx"`), and others (see their webpage for available solvers). Also available are a general nonlinear optimization using the augmented Lagrange multiplier method (`solver = "solnp"`) and a simple grid search (`solver = "grid"`). Lastly, setting `solver = c("grid", "solnp")` will first perform a coarse grid search and use the best solutions from this grid as starting values for repeated optimization with the solver (`"solnp"` in this example, but each available solver can be applied for this second step).

Advantages of Cognitivemodels

The **cognitivemodels** package is one way to facilitate the usage of formalized theories and to achieve robust cognitive modeling, which has been called for in recent meta-scientific proposals (e.g. Guest & Martin, 2020; Benureau & Rougier, 2018; Lee et al., 2019; Navarro, 2019; Van Rooij & Baggio, 2020; Wilson et al., 2017). Besides robustness of code, further advantages of the package are programming efficiency and flexibility.

Robustness. The models implemented in the **cognitivemodels** package are robust, because the package contains automated error checks and allows for manual error checks. By automated error checks we mean that there are unit tests implemented for the cognitive models in the package. Unit tests are automatic tests of parts of a program. Our unit tests test whether the model predictions are correct across a range of model parameters and input data. They also test if the default parameter estimation method replicates previously-obtained parameter values given published data. This is a safeguard against introducing programming errors during code development. By human error checks we mean that because the **cognitivemodels** package is open source, users of the package can report bugs through https://github.com/JanaJarecki/cognitivemodels/issues.

Programming Efficiency. The **cognitivemodels** package offers modelers a tool to fit cognitive models with minimal programming effort, because it uses a standardized syntax across the different cognitive models that are implemented in the package. The package requires minimal additional code to constrain parameters, make predictions, and compare models. The syntax of our package is similar to the syntax of the standard ANOVA or regression commands in R (e.g., the `predict()`, `anova()`, `logLik()`, or `coef()` methods). We believe, syntax standardization leads to efficiency gains when implementing models. Further, if standard models need not be implemented anew this saves implementation time.

Flexibility. The syntax for cognitive models in **cognitivemodels** allows modelers to adjust the modeling procedure to their own needs, to set parameter constraints, and to compare models. The package offers a general-purpose model development back-end in the R6 class which experienced modelers can use to implement further cognitive models in the package. This feature is not documented here, because of space limitations.

Summary

We have introduced **cognitivemodels**, the first R package to provide a robust, unified interface for formal modeling in cognitive science. Formalizing theories is seen as a crucial step to overcome the replication crisis in psychology. The **cognitivemodels** package may facilitate this step through its standardized and flexible syntax adapted to the needs of both novice and experienced modelers. We have exemplified the syntax of **cognitivemodels** with the generalized context model by applying basic and advanced modeling functionalities including model fitting with different types of parameter constraints and model testing with various

goodness of fit measures.

References

Andraszewicz, S. (2014). *Quntitative [i.e. Quantitative] analysis of risky decision making in economic environments* (PhD thesis).

Benureau, F. C. Y., & Rougier, N. P. (2018). Re-run, Repeat, Reproduce, Reuse, Replicate: Transforming Code into Scientific Contributions. *Frontiers in Neuroinformatics*, *11*(January), 1–8. http://doi.org/10.3389/fninf.2017.00069

Griffiths, T. L., Kemp, C., & Tenenbaum, J. B. (2008). Bayesian models of cognition. In R. Sun (Ed.), *The cambridge handbook of computational psychology* (pp. 59–100). Cambridge, UK: Cambridge University Press.

Guest, O., & Martin, A. E. (2020). *How Computational Modeling Can Force Theory Building in Psychological Science*.

Houston, A. I., & McNamara, J. M. (1988). A framework for the functional analysis of behaviour. *Behavioural and Brain Science*, *11*, 117–163. http://doi.org/10.1017/S0140525X00053061

Jarecki, J. B., Tan, J. H., & Jenny, M. A. (2020). A Framework for Building Cognitive Process Models. *Psychonomic Bulletin & Review*. http://doi.org/10.3758/s13423-020-01747-2

Juslin, P., Olsson, H., & Olsson, A.-C. (2003). Exemplar effects in categorization and multiple-cue judgment. *Journal of Experimental Psychology: General*, *132*(1), 133–156. http://doi.org/10.1037/0096-3445.132.1.133

Kass, R. E., & Raftery, A. E. (1995). Bayes Factors. *Journal of the American Statistical Association*, *90*(430), 773–795. http://doi.org/10.1080/01621459.1995.10476572

Lee, M. D., Chriss, A. H., & Vandekerckhove, J. (2019). Robust Modeling in Cognitive Science. *Computational Brain & Behavior*, *2*, 141–153. http://doi.org/10.1007/s42113-019-00029-y

Navarro, D. J. (2019). Between the Devil and the Deep Blue Sea: Tensions Between Scientific Judgement and Statistical Model Selection. *Computational Brain & Behavior*, *2*, 28–34. http://doi.org/10.1007/s42113-018-0019-z

Nosofsky, R. M. (1986). Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, *115*(1), 39–57. http://doi.org/10.1037/0096-3445.115.1.39

Nosofsky, R. M. (1989). Further tests of an exemplar-similarity approach to relating identification and categorization. *Perception & Psychophysics*, *45*(4), 279–290. http://doi.org/10.3758/BF03204942

Nosofsky, R. M. (2011). The Generalized Context Model: An Exemplar Model of Classification. In E. M. Pothos & A. J. Wills (Eds.), *Formal approaches in categorization* (pp. 18–39). Cambridge, UK: Cambridge University Press.

R Core Team. (2019). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.r-project.org/

Scheibehenne, B., & Pachur, T. (2014). Using Bayesian hierarchical parameter estimation to assess the generalizability of cognitive models of choice. *Psychonomic Bulletin & Review*, 391–407. http://doi.org/10.3758/s13423-014-0684-4

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*(2), 461–464. http://doi.org/10.1214/aos/1176344136

Singmann, H., & Kellen, D. (2013). MPTinR: Analysis of multinomial processing tree models in R. *Behavior Research Methods*, *45*(2), 560–75. http://doi.org/10.3758/s13428-012-0259-0

Tversky, A., & Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and Uncertainty*, *5*(4), 297–323. http://doi.org/10.1007/BF00122574

Van Rooij, I., & Baggio, G. (2020). Theory before the test: How to build high-verisimilitude explanatory theories in psychological science. Retrieved from https://psyarxiv.com/7qbpr/

Wagenmakers, E.-j., & Farrell, S. (2004). AIC model selection using Akaike weights. *Psychonomic Bulletin & Review*, *11*(1), 192–196. http://doi.org/10.3758/BF03206482

Wakker, P. P. (2008). Explaining the characteristics of the power (CRRA) utility family. *Health Economics*, *17*(12), 1329–1344. http://doi.org/10.1002/hec.1331

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLOS Computational Biology*, *13*(6), 1–20. http://doi.org/10.1371/journal.pcbi.1005510

Cognitive Flexibility in Cognitive Architecture: Simulating using Contextual Learning in PRIMs

Yang Ji (y.ji@rug.nl)
Jacolien van Rij (j.c.van.rij@rug.nl)
Niels A. Taatgen (n.a.taatgen@rug.nl)
Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence,
University of Groningen, Nijenborgh 9, 9747 AG Groningen, Netherlands

Abstract

The universal flexibility of biological systems needs to be reflected in cognitive architecture. In PRIMs, we attempt to achieve flexibility through a bottom-up approach. Using contextual learning, randomly firing of a set of instantiated primitive operators are gradually organized into context-sensitive operator firing sequences (i.e., primordial "skills"). Based on this implementation, the preliminary results of the model simulated the averaged single-pattern processing latency that is consistent with infants' differential focusing time in three theoretically controversial artificial language studies, namely Saffran, Aslin, and Newport (1996), Marcus, Vijayan, Rao, and Vishton (1999), and Gomez (2002). In our ongoing work, we are analyzing (a) whether the model can arrive at primordial "skills" adaptive to the trained tasks, and (b) whether the learned chunks mirror the trained patterns.

Keywords: cognitive flexibility; contextual learning; language acquisition; processing efficiency; PRIMs architecture

Introduction

From epigenetics to behavioral appropriateness, adaptability is ubiquitously observed at each level of biological systems (see Bateson & Gluckman, 2011). Cognition, in particular, may well be the most flexible system of all, which contrasts the deterministic approach in cognitive theories and modeling. To show that cognitive flexibility is possible, we use a generally-implemented model to simulate the learning of three specific language tasks by infants (for a review, see Saffran & Kirkham, 2018). The reasons are as follows. Firstly, it agrees with the consensus that language is one of the most crucial aspects of cognition (Newell, 1990; Rumelhart & McClelland, 1986). More importantly, the acquisition of language highlights the pivotal role of flexibility and adaptability. Last but not least, young infants cannot be instructed as how to acquire a language, therefore motivating a hard-code free approach.

In the following sections, an introduction of the three representative tasks is provided, before describing the common mechanism that learns all three tasks. We then compare the models' predictions to empirical data.

Three Language Phenomena

Without being endowed *a priori* with a native language, very young infants are sensitive to speech sounds (e.g., Kuhl, Williams, Lacerda, Stevens, & Lindblom, 1992), and are already discovering word forms within their first year of life (e.g., Jusczyk & Aslin, 1995). Such pioneering findings opened up a field focusing on infant language learning (see Saffran & Kirkham, 2018). However, it remains an open question as how infants can (a) identify atomic elements such as syllables (atomicity); and (b) compose atomic elements lexically and/or syntactically to form words or phrases (compositionality). This paper focuses on *compositionality* with the assumption of *atomicity*. In other words, it concerns the learning mechanism that connects lower-level syllables to higher-level words/phrases (Taatgen, 2017). This focus is discussed when the following tasks are introduced (see Figure 1).

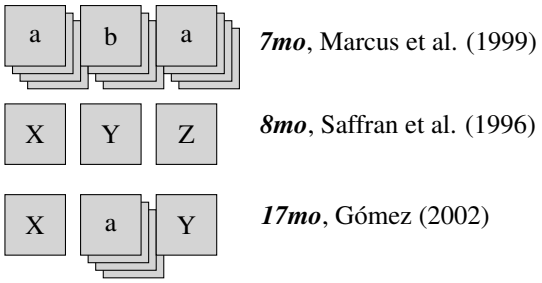


Figure 1: Three representative language tasks ordered based on developmental trajectory. Note: lowercase letter = variable token; uppercase letter = fixed token.

In Saffran et al.'s (1996) study, 8-month-olds are presented with an uninterrupted speech stream formed by randomly concatenating four fixed trisyllabic words in the form of *X-Y-Z* (see Figure 1). After the learning phase, infants are examined with a set of test words. Infants show more attention to novel non-word (e.g., "da-pi-ku") or part-words ("*tu*-da-ro") as compared to test words directly taken from the training phase (e.g., "da-ro-pi"). Saffran and colleagues (1996) interpreted their results from a connectionist perspective. They considered that infants' differentiation of speech streams is related to the acquirement of embedded transition probabilities between adjacent word-syllables (*statistical learning*). Nevertheless, the differentiation of speech streams at the global level does not fully explain whether word forms are learned/segmented. To verify this further, in a follow-up study, 17-month-old infants performed a label-object association task after listening to a continuous stream of words (Estes, Evans, Alibali, & Saffran, 2007). During

the habituation phase, shapes (i.e., objects) are either presented with words or other untrained non-words/part-words (i.e., labels) until a habituation criteria is reached. In the test phase, object-label pairings are switched to induce dishabituation. However, only when the labels are words versus non-word/part-words was the dishabituation detected, which implies that wordlike units are necessary for label-object association, and needs to be learned during the training phase. Similarly in modeling studies, previous simulated results were more in line with empirical findings when both token-level transitional probabilities and the generation of word-level patterns were taken into consideration (see Mareschal & French, 2017). These altogether support that infants are able to *compose* atomic elements into basic lexical units.

In contrast, Marcus et al. (1999) argued that pure connectionist learning account may not be applicable in all situations, and that syntactic structure is needed to recognize generalized pattern types. They showed that 7-month infants seem to be able to derive the more general *a-b-a* pattern, after being presented with a series of trisyllabic patterns of "le-we-le", "ga-ka-ga" and so on (see Figure 1). Infants focus distinctively more on the novel test patterns of *c-d-d* and *c-c-d*, as compared to the familiarized test pattern of *c-d-c*, even when the specific tokens are replaced. Marcus et al. (1999) therefore showed that infants are able to generalize even though there exist no transitional probabilities between the learned and test patterns (*algebraic learning*). However, their argument that infants possess innate ability of syntactic processing (e.g., knowing "the 1st token predicts the 3rd" in *a-b-a*) is not in line with empirical findings. In fact, infants generally needs to be more than 1-year-old to distinguish non-adjacent syntactic relations (e.g., Gómez & Maye, 2005). Alternatively, more recent studies have shown that younger infants (7-month-olds) are instead attuned to immediate repetitions without being able to acquire the full trisyllabic pattern (e.g., Wagner, Fox, Tager-Flusberg, & Nelson, 2011). Our previous model shows that the learning and transfer of algebraic patterns can be achieved in a bottom-up fashion when immediate repetition are rewarded (Ji, van Rij, & Taatgen, 2019). Therefore, the findings of Marcus et al. (1999) may only captures infants' ability to identify a particular element as it is (i.e., *atomicity*), rather than the capability to understand syntax fully. However, this is not to say that infants cannot learn syntactic structures. Research in syntactically-relevant non-adjacent dependency learning is championed by Gómez and colleagues. Taken non-adjacent pairs in the form of *X-a-Y* as an example (e.g., "pel-*a*-rud", see Figure 1), when variability of the middle token *a* (i.e., 24 variations) renders transitional probabilities unreliable to capture the regularity of that pattern, 17-month-old infants counterintuitively are better able to differentiate them by focusing more on novel non-adjacent pairs ("pel-*a*-jic") than learned ones (Gomez, 2002). Infants are therefore able to shift strategies (Saffran & Kirkham, 2018, p. 190), suggesting diverse form of language compositionality either by *lexicon* or *syntax*.

One Architecture that Learns

Although the theory of artificial language learning remains controversial, it is indisputable that infants have the ability to deal with all tasks. However, usually in the ACT-R model, stimulus-response production-rules related to task processing need to be artificially defined. Thus, the discovery and learning process of infants cannot be well simulated. For the problem of skill acquisition, Taatgen and Lee (2003) first proposed the learning strategy of production compilation and incorporating it into ACT-R. Through production compilation, general production-rules are combined into task-specific rules adapted to the task-demand. As early as 2002, Taatgen and Anderson (2002) boldly applied procedurally-related production compilation in children's language learning, and shows how regular past-tense rule can be learned as a specialization of more general rules. Until recently, the procedural hub of basal ganglia is viewed as relevant not only to motor learning, but also to many other skill domains including language (see Stocco, Lebiere, & Anderson, 2010, Kotz & Schmidt-Kassow, 2015). Nevertheless, the firing conditions (i.e., context or goal-state) and information processing flow of general-purpose production rules still need to be programmed manually. Moreover, production compilation is operated at the procedural level, but learned skills are often transformed as long-term declarative knowledge that can be transferred/reused in different scenarios (see Stocco et al., 2010).

For the same question of incorporating skill acquisition in a cognitive architecture, we propose a new bottom-up approach that seeks to organize primitive elements of procedural knowledge into context-sensitive stimulus-response rules through trial-and-error. These rules are maintained in declarative memory and can be transferred in other task contexts once necessary. The specific contextual learning mechanism to achieve this is inspired by the action selection process of basal ganglia and related cortical areas (see Stocco et al., 2010, Dehaene, Meyniel, Wacongne, Wang, & Pallier, 2015). The basal ganglia is a reinforcement learning hub that synthesizes contextual signals from multiple cortical areas and connects them with corresponding responses, and relays response outcomes gradually to the cortex to be maintained and integrated with contextual information. When a task-related reward state is reached, the cortex then fine-tunes the association between the contexts and specific primitive procedural elements to promote the rearrival at such task-relevant reward state. For infants' performance on artificial language tasks, Saffran and Kirkham (2018, p. 195) similarly suggests that reinforcement learning maybe a crucial candidate for language acquisition. It is possible that such reinforcement learning of language skills is supported by the cortico-basal ganglia mechanism (Kotz & Schmidt-Kassow, 2015).

In addition, the fine-tuning of contextual learning requires predefined reward states. Empirical evidence for these states are provided by Wagner et al. (2011). It is found that younger infants are more susceptible to the changing environment, especially the *exogenous* repetition of simple stimuli. For

slightly older infants, these simple environment-driven reactions are gradually replaced by the *endogenous* detection of more complex embedded regularities that are mirrored in memory. The learning from simple elements to more complex patterns is also reflected in animal studies. In one study, a saccade task with four targets is presented to the macaques. At the beginning of training, basal ganglia and related cortical areas respond to all single targets. However, cellular response in later stages is limited to only the sequential boundary made up of the four targets (see Dehaene et al., 2015, p. 5). These results support that the basal ganglia-inspired contextual learning mechanism may be the key to the transition from atomicity to compositionality.

The purpose of this article is to provide a proof of concept for the bottom-up learning approach. Through contextual learning, we investigate whether the model can provide a unified description on three theoretically controversial artificial language tasks, namely Saffran et al. (1996), Marcus et al. (1999), and Gomez (2002). In this article, our first task is to simulate and explain the experimental results, that is, focusing time differences. Currently, we are still analyzing the procedural and declarative knowledge acquired by the model under different task conditions.

Model

The model is implemented in the PRIMs architecture (see Taatgen, 2013). In PRIMs, operators are equivalent to production-rules, but with slightly different nature (van der Velde, 2018). Like ACT-R, operators are if-then rules that define how information are routed and compared between perceptual and memory buffers. Moreover, these operators can be further broken down into their smallest units (i.e., primitive operations). Contrary to ACT-R, in PRIMs operators share the properties of chunks, including base-level activation and spreading activation from the buffers. This is because procedural operations will eventually be stored in cortex to be used in future scenarios (Stocco et al., 2010, p. 548). Therefore, operators can be triggered based on its associations with the current buffer contents (i.e., the immediate contexts). For example, and operator can be triggered by a certain auditory input, or by a previously executed operator. Associations between the operators and the contexts are learned through reinforcement learning. This partially replaces goal-states that used to be explicitly defined for action selection with production-rules. The gradual acquisition of *context-sensitive operations* increases the flexibility of the architecture, and opens up a method of exploration-based learning.

Primitive Operations Primitive operations are the smallest units of production rules. They route and compare information between different buffers. In this model, environmental inputs can be encoded successively to the slots of the imaginal buffer. For example, the previous stimulus X fills the currently empty slot (e.g., slot-2, if slot-1 is filled) in the imaginal, the next stimulus Y can only fill the next free slot (e.g., slot-3). Encoding of the environmental stimulus in the imaginal also automatically starts the retrieval of the declarative memory chunk containing the stimulus, and the chunk with the highest activation and exceeding the retrieval threshold can be harvested. When the imaginal buffer is cleared (in PRIMs, this is achieved when "nil" is filled into imaginal slot-0), the chunk stored in the current imaginal will then be stored in the declarative memory as a whole.

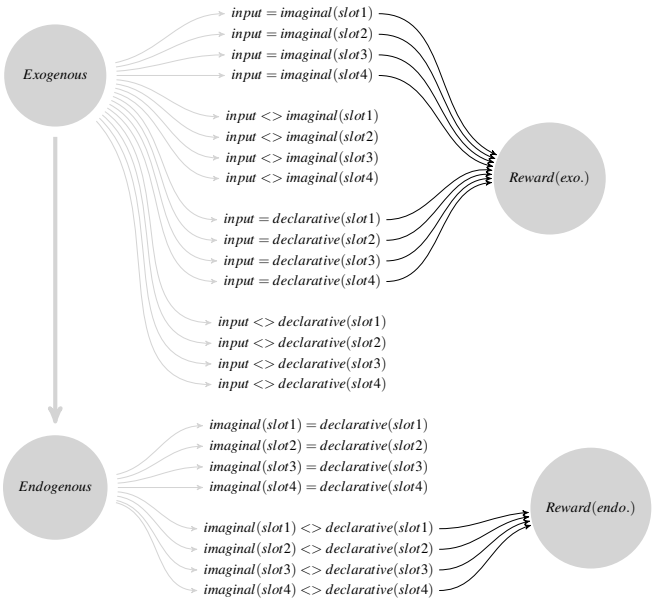


Figure 2: Comparison operations and reward preferences. Note: exogenous = comparisons with input; endogenous = comparisons with declarative unit; Reward(exo./endo.) = Exogenous/endogenous reward preferences.

When memory buffers and/or input buffers are not empty, another series of comparison operations can be fired to check whether there is match/mismatch between the buffer slots. Comparison operations in this model are categorized into *exogenous* and *endogenous* types (see Wagner et al., 2011). *Exogenous* operations check whether the immediately presented stimuli matches/mismatches the slot in the chunk currently stored in imaginal or retrieved from declarative memory. The reward state is to detect any immediate matches between input and the memory buffers (i.e., *Reward(exo.)*, Figure 2). *Endogenous* operations check whether the currently encoded pattern by slot mirrors the pattern as retrieved from declarative memory. The reward state is to find a mismatch that identifies the pattern boundary (i.e., *Reward(endo.)*, Figure 2).

A Walk-Through Example Here, we describe one possible processing solution to the specific pattern of "le-we-le". Suppose the model has already learned the bigram "le-we" in declarative memory. When the first input "le" is encoded into imaginal slot-1, the automatic retrieval process may harvest "le-we". Consequently, with the encoding of the second input "we", the model may find the syllable is now matched between slot-2 of the imaginal and the declarative chunk. How-

ever, after the encoding of the third input "le", a mismatch maybe found between slot-3 of the imaginal and declarative chunk (i.e., an empty slot), which suggest the current presented pattern are different from the memorized pattern at the global level. This time, the endogenous reward state is arrived. If there is sufficient time the model will strengthen associations between all fired operations and their corresponding contextual buffer states that lead to the reward.

On the other hand, the model may also process the pattern in an exogenous manner, and just find that the encoded first syllable "le" in imaginal slot-1 is repeated when the third "le" is presented. If the operations and related buffers states are reinforced after this reward state is reached, the model may alternatively oriented towards the detection of immediate repetition.

Learning Mechanism The learning mechanism binds operations to their contexts in accordance with the reward preference. When a particular reward state is satisfied, the associations between operations and the contexts are strengthened:

$$\Delta S_{jik} = \beta \, (\, \textbf{payoff} - S_{jik} \,) \tag{1}$$

where

$$\textbf{payoff}_{\,fired} = \frac{\max S_{jik} \times (\text{ reward - timeToReward })}{\text{reward}} \tag{2}$$

At the same time, the bond between unused operations and the contexts are weakened by the payoff term:

$$\textbf{payoff}_{\,unfired} = \frac{\max S_{jik} \times (\text{ 0 - timeToReward })}{\text{reward}} \tag{3}$$

In this function, association weight S_{jik} is updated every time when an reward is issued. Here, j denotes the specific operation fired, and ik denotes the associated context in buffer i and slot number k. The *beta* is a learning rate parameter, whereas the *payoff* term specifies how much context-operation association weights are updated each time. Specifically, *timeToReward* is the firing time of each used operation. *Reward* is the sum of the set reward parameter ($Reward_0$ = 10.0) and the "trial" duration (i.e., previous to current reward time). After the rewards are issued, the imaginal is cleared, and the next "trial" now starts afresh.

Timing Consideration From birth to the age of 2 years, processing efficiency undergoes dramatic age-related changes without altering structure of the brain (see Dubois, Adibpour, Poupon, Hertz-Pannier, & Dehaene-Lambertz, 2016). This means that young infants are only able to fully process a stimulus when presentation is sufficiently long (see Chen, Peter, & Burnham, 2016). In the current model, the interaction between stimulus duration and the rate of operation firing is specifically considered. Operation firing takes time, and when the stimulus duration is short, the per-stimulus operation firing rate will be reduced accordingly. In addition, if the operator processes have not yet ended when the presentation of the current stimulus has ended, the processing time of the operation would therefore exceed the presentation of the stimulus. In this case, the processes of the upcoming stimulus would be less sufficient as if its presentation time has been reduced. In this model, it is so far arbitrarily set that the stimulus would be completely ignored when the time window for processing a stimulus is reduced to less than 10% of the objective stimulus presentation time.

Object of Evaluation Based on the general implementation, it is investigated whether infants' differential focusing time for different task conditions can be simulated by a single model. Our simulation and experimental results apply different time scales. The experimental results investigate the overall focusing time for all patterns in the test phase, whereas the simulation results focusing on the averaged time required to process a single pattern in the test phase. The reasons are as follows. First of all, we do assume that processing latency of a single pattern is related to the overall focusing time. When the infants need to spend more time to process a pattern, then the remaining task-irrelevant gap will be shortened accordingly. In this case, the probability of the infants deviating from the task is relatively small, so the overall focusing time will be relatively high. However, if the infants are now familiar with the pattern and can effectively process it, task-irrelevant gap will increase. At this time, the possibility that the infants deviate from the task will also increase, resulting in a decrease in the overall focusing time. However, for the current model, we do not know what the cause and duration when infants divert from the task. There are many possibilities, such as when an infant is captured by other interesting environment stimuli (external causes), or the needs for food or play (internal causes). It is much more difficult to reflect these factors in the current model. Therefore, we only consider the single-pattern processing latency of the learned and novel patterns after training, and assume the difference in processing latency would reflect the overall difference in focusing time. Specifically, duration from each stimulus onset to its last operation firing time are summed and averaged for each pattern. Note that when the operation/s cross the next stimulus boundary, the stimulus input onset will be delayed. In the next section, details of each task conditions are described, followed by the simulated results.

Experimental Details

Saffran et al. (1996) In this task, infants are first presented with a training stream of continuous trisyllabic patterns. These trisyllabic patterns include four words in the form of *X-Y-Z* (i.e., "pa-bi-ku", "ti-bu-do", "da-ro-pi", and "go-la-tu"). These four words are concatenated randomly with no interval between them. After training, the experiments 1 and 2 further test whether infants exhibit a difference in the duration of focusing time on trained words versus untrained patterns during the test phase. The tested trained words are directly taken from two words presented during the training phase (i.e., "pa-bi-ku", and "ti-bu-do"), whereas the structure of untrained patterns and trained words have differ-

ent transitional probabilities to the trained words. In Experiment 1, the untrained non-words (i.e., "da-pi-ku", "ti-la-do") share no transitional probabilities ($p = 0$) with the trained words. In Experiment 2, the untrained part-words (i.e., "*tu*-da-ro", and "*pi*-go-la") share some transitional probabilities ($p = \frac{1}{3}$) with the trained words, as if the part-words are crossing over the word boundaries.

Marcus et al. (1999) The study investigated whether after training a certain type of trisyllabic patterns, infants show differential focusing time to the tested same and different pattern types with replaced tokens. In other words, the trained and tested patterns share no transitional probabilities. In the original experiment, the stimulus properties of experiments 2 and 3 are better-controlled. In Experiment 2, the training patterns are of the type *a-b-a* or *a-b-b*, and the test pattern types contain *c-d-c* and *c-d-d*. Similarly, in Experiment 3, the training patterns are of the type *a-b-b* or *a-a-b*, and the test pattern types are *c-d-d* and *c-c-d*. In this study, results regarding consistent and inconsistent type to the trained pattern are collapsed together. For all training patterns, there are four instantiations relates to the repeated syllables (**a**-*b*-**a**, a-**b**-**b**, or **a**-**a**-*b*; i.e., "le", "wi", "ji", and "de"); and two instantiations for non-repeated syllables (*a*-**b**-*a*, **a**-*b*-*b*, or *a*-*a*-**b**; i.e., "di", "je", "li", and "we"). For the patterns in the test phase, the instantiations of repeated (**c**-*d*-**c**, c-**d**-**d**, and **c**-**c**-*d*; i.e., "ba", "ko") and non-repeated syllable (*c*-**d**-*c*, **c**-*d*-*d*, and *c*-*c*-**d**; i.e., "po", "ga") are replaced.

Gómez (2002) The study investigated whether after training non-adjacent dependent *X-Y* patterns, infants show differential focusing time on the same *X-Y* pattern from other untrained *X'-Y'* non-adjacent dependent patterns. Here, the meaning of non-adjacent dependency is that the fixed tokens *X* and *Y* are not adjacent to each other, but separated by a variable token *a* in the form of *X-a-Y*. In the training phase, *X-Y* has two instantiations (i.e., "pel-rud", "vot-jic"). They are divided into three conditions based on the variability of the middle token *a*. In the first condition, *a* has only 3 variants, while in the second and third conditions, the variability of *a* increases to 12 and 24 variants. The instantiation of *a* includes "wadim", "kicey", "puser" and so on. During the test phase, it is then investigated whether infants can distinguish the same *X-Y* and different patterns *X'-Y'* after each of the three training conditions. The same test *X-Y* patterns are consistent with the training phase (i.e., "pel-rud", "vot-jic") and are separated by a variable token *a* that includes three variants. Similarly, the different test *X'-Y'* patterns uses the reverse of *X-Y* (i.e., "pel-jic", "vot-rud"), which also includes a 3-variant middle token *a*.

Task Lengths In the empirical studies, the presentation time of trisyllabic patterns in different tasks varies greatly. Specifically, in Saffran et al. (1996), the presentation time of each trisyllabic pattern is 750 ms (250 ms/syllable) without inter-pattern interval (note though that a 500 ms inter-pattern interval is added during the test phase). In Marcus et al. (1999) and Gomez (2002), trisyllabic pattern are each presented for 1500 ms (500 ms/syllable) with an inter-pattern interval of 1000 ms and 500 ms respectively. In other words, the numbers of patterns trained in the Saffran et al. (1996) greatly exceeded the other two tasks. In the simulation study, in order to make the model fully acquire the patterns in each tasks, the simulation duration is set longer than it is in the empirical study. In detail, the total presentation lengths in the training phases are 500 patterns (6.25 min) for Saffran et al. (1996, 2 min), 100 patterns (4.17 min) for Marcus et al. (1999, 2 min), and 100 patterns (3.33 min) for Gomez (2002, 3 min). In addition, to better compare the operation firing patterns in the early and late training stages, the model divides the entire continuous presentation of patterns into streams. Each stream contains the continuous presentation of 10 patterns. On the other hand, in the test phase of the simulation, all task conditions consistently contains 10 patterns. This is done so because our model only focuses on how much time is spent on average to process a single trisyllabic pattern (along with the immediate inter-pattern interval that follows).

In addition, our model assumes that processing efficiency undergoes change during different months of age. Infants of 7-8 months are tested in Saffran et al. (1996) and Marcus et al. (1999), while 17-month-olds are tested in Gomez (2002). In our model, processing efficiency is differentiated by the firing duration of an individual operator, and operations that are not successfully fired also take time. To simulate younger infants in Saffran et al. (1996) and Marcus et al. (1999), the firing duration is set lower (70 ms) as compared to older infants (50 ms) in Gomez (2002).

Simulation results

Learning of Context-Sensitive Operation In here, we only show the difference between the operation firing responding to each single pattern of various task conditions during training (see Figure 3). For the specific firing pattern formed by these operations, we are currently conducting further analysis and will not be elaborated in this paper. Initial and later state describe the performance of the model in the first stream and the tenth stream respectively. Note that the number of streams applied here is only for demonstration purpose and do not represent the entire training length - for example, simulation of Saffran et al. (1996) consists of 50 streams (500 continuous patterns). We can see that in the initial state, the firing of the operation is without structure. However, in the later state, the operation seems to form some firing patterns. In addition, the efficiency of firing seems to have improved, so it can be seen that the transitional gaps between stimuli and/or pattern are also increased. However, for the simulation of Saffran et al. (1996), the increase in transitional gap is not as obvious. This is because in this experiment, the presentation time of each syllable stimulus is extremely short (250 ms) and there is no inter-pattern interval between patterns during the training phase.

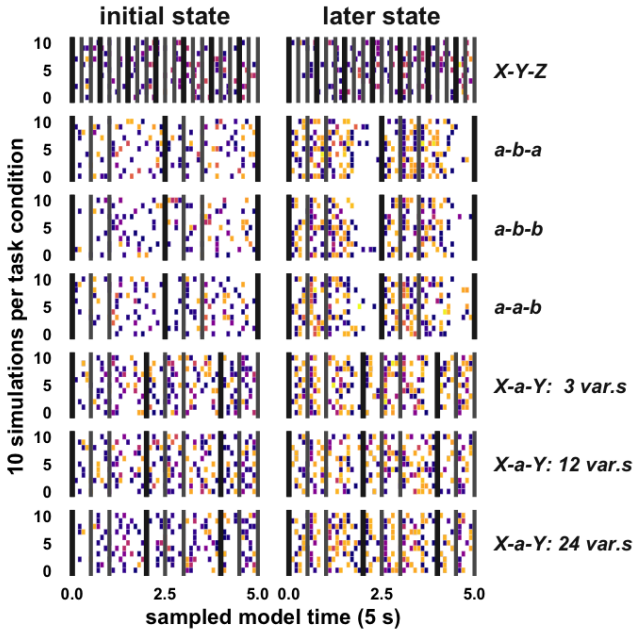


Figure 3: Changes in operator firing for different tasks (grids represents different tasks; scales from 0 to 10 on the grid-Y-axis represents the 10 simulated subjects sampled) and different parts of the training (left blocks are onsets of training, right blocks are later stage in training). Each dot represents a fired operator, with varying colors for different operator types. The black vertical lines mark the onsets of sound patterns, the gray lines mark the onset of individual syllables.

Differentiating Acquired/Novel Patterns The empirical and the stimulated results are analyzed based on an unpaired two sample design. The reason for this is that (a) the experimental results leave only summarized data (such as mean, standard error, and sample size), therefore the original within-group difference cannot be reassessed; whereas (b) in the model, different task conditions are independently simulated. Moreover, Welch's t-test is performed to analyze the results, since (a) the original data is based on small samples, and (b) we do not assumed equal variance in experimental and simulated samples. For **Saffran et al. (1996)**, it was found that none of the experiments' focusing time difference on words/non-words (*mean difference* = 0.88 s, $p = 0.16$) and words/part-words (*mean difference* = 0.83 s, $p = 0.17$) reached statistical significance (see Figure 4A). Similarly in simulation, no difference was found between words/non-words (*mean difference* = 0.002 s, $p = 0.61$) and words/part-words (*mean difference* = 0.006 s, $p = 0.12$) for the single-pattern processing latency (see Figure 4B). Nevertheless, regardless of experimental and stimulated results, it is found, at face level, that the focusing time and processing latency for trained words is slightly longer than non-words/part-words. **Marcus et al. (1999)** investigated the focusing time difference between acquired/novel pattern types, and found that

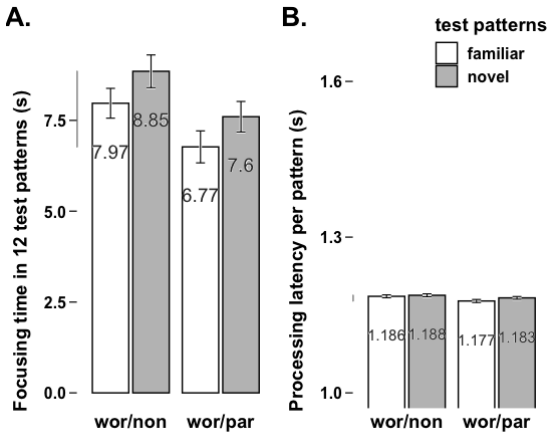


Figure 4: Simulation of Saffran et al., 1996. *A:* Data, average focusing time (±1SE) in 12 patterns. *B:* Simulation, processing latency *per pattern* (average of 200 runs, ±1SE). Note: wor = words; non = non-words; par = part-words.

there was significant differences between cdc/cdd (*mean difference* = 1.75 s, $p = 0.04$) and cdd/ccd (*mean difference* = 2.00 s, $p = 0.003$; see Figure 5A). Similarly in simulated results of single-pattern processing latency, we also found the difference between cdc/cdd (*mean difference* = 0.10 s, $p = 2.63 \times 10^{-9}$, *Cohen's d* = 0.61) and cdd/ccd (*mean difference* = 0.07 s, $p = 1.98 \times 10^{-4}$, *Cohen's d* = 0.38; see Figure 5B). Analysis of **Gomez (2002)** shows that the greater the variability of middle token *a* during the training phase, the larger the focusing time difference between acquired non-adjacent patterns and novel patterns. However, only when the variability contains 24 instantiations (*mean difference* = 0.05 s, $p = 0.97$), the focusing time difference reaches significance; whereas when the variability is with 3 (*mean difference* = 0.34

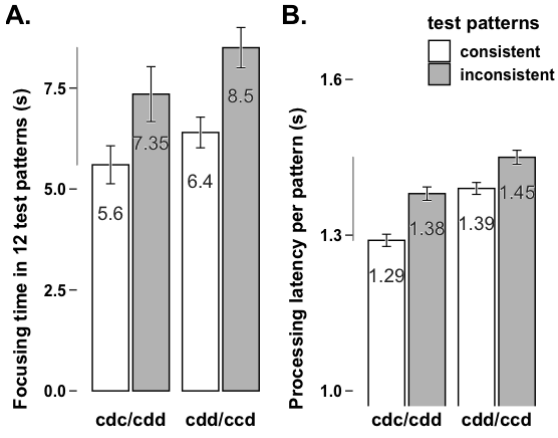


Figure 5: Simulation of Marcus et al., 1999. *A:* Data, average focusing time (±1SE) in 12 patterns. *B:* Simulation, processing latency *per pattern* (average of 200 runs, ±1SE).

s, $p = 0.73$) and 12 instantiations (*mean difference* = 2.07 s, $p = 0.003$), the differences are non-significant (see Figure 6A). In the simulated results due to the larger sample size, the differences in single-pattern processing latency among 3 (*mean difference* = 0.04 s, $p = 0.007$), 12 (*mean difference* = 0.06 s, $p = 6.79 \times 10^{-5}$), and 24 instantiation conditions (*mean difference* = 0.07 s, $p = 8.30 \times 10^{-7}$) have all reached significance. Further analysis of the simulated results shows that effect sizes increases as the number of instantiations increases from 3 (*Cohen's d* = 0.27), 12 (*Cohen's d* = 0.40) to 24 (*Cohen's d* = 0.501). Only the 24-variant condition shows substantive medium effect size latency difference (see Figure 6B).

Discussion

In this study, we use a single model to simulate three theoretically controversial infant artificial language tasks. The simulated results of different tasks are consistent with the original findings. Specifically, for Marcus et al. (1999), simulated difference in processing latency is found between consistent/inconsistent pattern types after training; and for Gomez (2002), as the variability of token *a* in non-adjacent dependent pattern *X-Y* increases, the difference in processing latency between trained/novel patterns gradually increases, showing substantive difference only when token is instantiated with 24 variants. These simulated results indirectly illustrate enhanced processing efficiency of the learned pattern. This is assumed to reserve longer task-irrelevant gap during pattern processing, thereby increasing the possibility of diversion and eventually leading to a reduction in focusing time for the trained pattern. Therefore, the simulated results are consistent with empirical findings and illustrates the learning ability of the model.

Nevertheless, for Saffran et al. (1996), further analysis suggests that the original data or the simulated results only revealed face level difference but neither reached statistical significance. This is the case even though the simulated length of this task is the longest. In Saffran et al. (1996), the presentation time of each syllable in the pattern is only 250 ms and without inter-pattern interval. Therefore, it is difficult for infants to sufficiently process the patterns. For example, for the trained pattern of "da-ro-pi", it is very likely that infants may only process "da" and "pi" but omit the middle syllable "ro". In addition, the model's reinforcement learning process also takes time, and the pattern presentation time is thus too short and prevents such reward process from occurring. These are among the reasons that the operation firing pattern are still sparse at the end of training (e.g., Figure 3). Our ongoing analysis did find that the operation firing patterns are differentiated for the trained/novel patterns. Though due to syllable omission, the model tends to acquire skip-grams rather than trigrams.

In general, our simulation shows that the model can gradually acquire the different task patterns through a cognitively constrained architecture, avoiding views that consider task-specific information processing as innate and deterministic.

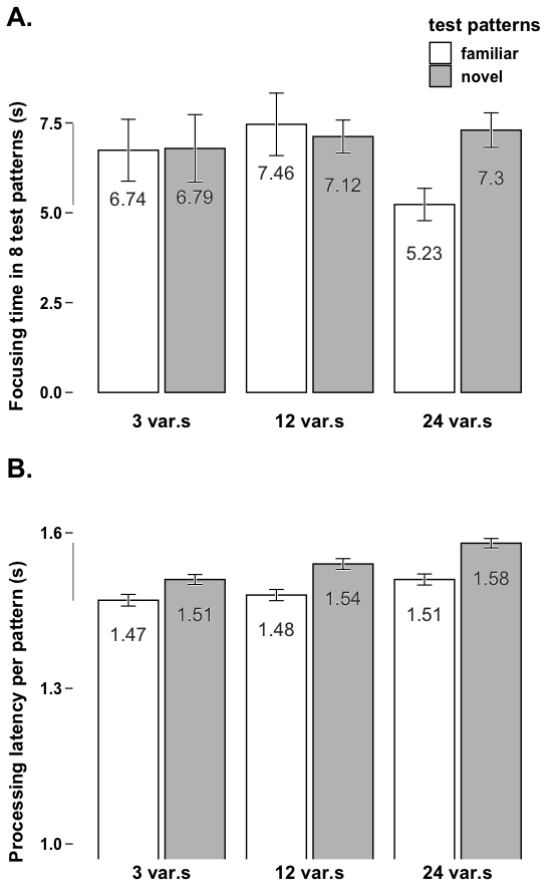


Figure 6: Simulation of Gómez, 2002, who tested the differentiation of *X-Y* and *X'-Y'* patterns after training *X-Y* with 3, 12, and 24 variations of middle token *a*. *A:* Data, average focusing time (±1SE) in 8 patterns. *B:* Simulation, processing latency *per pattern* (average of 200 runs, ±1SE).

Conclusion

The current simulation provides unified descriptions for the three artificial language tasks. The model can distinguish between task conditions at the level of processing latency, which implies its capabilities to acquire operation firing patterns or primordial "skills" related to the task conditions without explicit programming. Therefore, the PRIMs contextual learning mechanism contributes to the flexibility of cognitive architecture. However, to tackle the question of compositionality, this article has a few limitations and remains incomplete. The simulation has not considered the complex factors that lead to diversion, and the overall focusing time of the entire test phase has not been simulated. In addition, we are still analyzing the procedural and declarative knowledge acquired by the model. Only by answering this question can we better demonstrate the skill acquisition of PRIMs and the sure acquirement of language-related contents.

References

Bateson, P., & Gluckman, P. (2011). *Plasticity, robustness, development and evolution*. NY: Cambridge University Press.

Chen, A., Peter, V., & Burnham, D. (2016). Auditory erp response to successive stimuli in infancy. *PeerJ*, *4*, e1580.

Dehaene, S., Meyniel, F., Wacongne, C., Wang, L., & Pallier, C. (2015). The neural representation of sequences: from transition probabilities to algebraic patterns and linguistic trees. *Neuron*, *88*(1), 2–19.

Dubois, J., Adibpour, P., Poupon, C., Hertz-Pannier, L., & Dehaene-Lambertz, G. (2016). Mri and m/eeg studies of the white matter development in human fetuses and infants: review and opinion. *Brain Plasticity*, *2*(1), 49–69.

Estes, K. G., Evans, J. L., Alibali, M. W., & Saffran, J. R. (2007). Can infants map meaning to newly segmented words? statistical segmentation and word learning. *Psychological Science*, *18*(3), 254–260.

Gómez, R., & Maye, J. (2005). The developmental trajectory of nonadjacent dependency learning. *Infancy*, *7*(2), 183–206.

Gomez, R. L. (2002). Variability and detection of invariant structure. *Psychological Science*, *13*(5), 431–436.

Ji, Y., van Rij, J., & Taatgen, N. A. (2019). Discoveries of the algebraic mind: A prims model. In T. Stewart (Ed.), *Proceedings of the 17th international conference on cognitive modeling* (pp. 71–76). Waterloo, Canada: University of Waterloo.

Jusczyk, P. W., & Aslin, R. N. (1995). Infants detection of the sound patterns of words in fluent speech. *Cognitive Psychology*, *29*(1), 1–23.

Kotz, S. A., & Schmidt-Kassow, M. (2015). Basal ganglia contribution to rule expectancy and temporal predictability in speech. *Cortex*, *68*, 48–60.

Kuhl, P. K., Williams, K. A., Lacerda, F., Stevens, K. N., & Lindblom, B. (1992). Linguistic experience alters phonetic perception in infants by 6 months of age. *Science*, *255*(5044), 606–608.

Marcus, G. F., Vijayan, S., Rao, S. B., & Vishton, P. M. (1999). Rule learning by seven-month-old infants. *Science*, *283*(5398), 77–80.

Mareschal, D., & French, R. M. (2017). Tracx2: a connectionist autoencoder using graded chunks to model infant visual statistical learning. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *372*(1711), 20160057.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*. Cambridge, MA: MIT Press/Bradford Books.

Saffran, J. R., Aslin, R. N., & Newport, E. L. (1996). Statistical learning by 8-month-old infants. *Science*, *274*(5294), 1926–1928.

Saffran, J. R., & Kirkham, N. Z. (2018). Infant statistical learning. *Annual Review of Psychology*, *69*, 181–203.

Stocco, A., Lebiere, C., & Anderson, J. R. (2010). Conditional routing of information to the cortex: A model of the basal ganglia's role in cognitive coordination. *Psychological Review*, *117*(2), 541.

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471.

Taatgen, N. A. (2017). Cognitive architectures: Innate or learned? In *A standard model of mind: Technical report fs-17-05* (p. 476—480). Association for the Advancement of Artificial Intelligence.

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say "broke"?: A model of learning the past tense without feedback. *Cognition*, *86*(2), 123–155.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors*, *45*(1), 61–76.

van der Velde, M. A. (2018). *Modelling the effect of depression on working memory*. Unpublished master's thesis.

Wagner, J. B., Fox, S. E., Tager-Flusberg, H., & Nelson, C. A. (2011). Neural processing of repetition and non-repetition grammars in 7-and 9-month-old infants. *Frontiers in Psychology*, *2*, 168.

Implementing Incentive-Sensitization Theory of Addiction with Nengo Neural Network Simulator

Peiying Jian (peiying.jian@queensu.ca)[1]
Terrence C. Stewart (terrence.stewart@nrc-cnrc.gc.ca)[2]
Mary C. Olmstead (olmstead@queensu.ca)[1]
[1]Department of Psychology, Queen's University
403 Craine, Kingston, ON K7L 3N6 Canada
[2]National Research Council of Canada, University of Waterloo Collaboration Centre
200 University Avenue West, Waterloo, ON N2L 3G1 Canada

Abstract

We present first steps towards a biologically grounded implementation of the Incentive-Sensitization Theory of addiction. We present multiple different plausible ways of mapping this theory into a computational model, and examine the resulting behaviour to see whether it accords with standard interpretations of the theory. This is the first step in a larger project to create a computationally tractible and biologically motivated model of addiction to help clarify and ground various terms in the theory.

Keywords: Addiction; Computational Modelling; Incentive Sensitization

Background

Prior to the 1980s, addiction was often viewed as a character flaw or personal failure (Frank & Nagel, 2017). This moral model of addiction assumes that drug use is voluntary: addicts consciously choose to self-administer drugs despite knowledge of adverse personal and societal costs associated with this behaviour. A growing understanding of the biological consequences of excessive drug use helped to establish the medical model of addiction, where behavioural patterns associated with maladaptive drug use (e.g., compulsion, impulsivity, craving, relapse) are driven by underlying changes in the structure and function of neural circuits mediating reward, motivation, and decision making (Koob & Volkow, 2016).

The medical model of addiction builds on positive reinforcement theories, proposing that drug-taking is reinforced by the euphoric state of drug use. These effects are mediated through the mesolimbic dopamine (DA) pathway, which projects from the ventral tegmental area to the nucleus accumbens and prefrontal cortex (Wise, 1980). Antagonizing mesolimbic DA activity reduces the reinforcing effects of both abused drugs and natural reinforcers like food or sex (Koob & Le Moal, 1997; Wise, 1980; Farooqi et al., 2007).

The medical model of addiction acknowledges that negative reinforcement also contributes to continued drug use in that drug intake alleviates negative symptoms of withdrawal, such as drowsiness, headache, or depression. With chronic use, mesolimbic DA system function is altered (Pierce & Kumaresan, 2006) such that baseline DA levels are decreased in drug-free states. As a consequence, substance abusers experience less drug-induced pleasure as addiction develops; they also increase the dose and frequency of drug intake to make up for reductions in baseline DA levels (Koob, 2020).

The Incentive-Sensitization Theory of Addiction

By the mid 1990s, it became clear that neither positive nor negative theories of reinforcement provided a full account of addictive behaviour. For example, the positive reinforcing effects of natural reinforcers, such as food or sex, are also mediated by the mesolimbic DA system but individuals are less likely to develop compulsive intake of these commodities. Negative reinforcement also fails to sufficiently explain addiction. Psychostimulants, such as cocaine, are clearly addictive but do not produce strong somatic, withdrawal symptoms. In addition, intensified withdrawal symptoms do not elicit robust drug craving (Shaham, Rajabi, & Stewart, 1996). Therefore, although both positive and negative reinforcement may contribute to continued drug use, they cannot explain fundamental aspects of addictive behaviour.

In response, Robinson and Kolb (1999) formulated the Incentive-Sensitization Theory of drug addiction which proposes that 'wanting' and 'liking' drugs are mediated by two different mechanisms. In support of this theory, 'liking' in rats, assessed as orofacial responses to presentation of a sweet solution, is unaffected by depletion of mesolimbic DA whereas the same manipulation reduces motivation to obtain a reward (i.e., 'wanting') (Berridge, Venier, & Robinson, 1989). Subsequent work confirmed a dissociation in biological systems that mediate these two processes in rodents (K. C. Berridge, 2007; Pool, Sennwald, Delplanque, Brosch, & Sander, 2016). Similarly, in human patients with reduced DA function (Parkinson's disease), ventral striatal DA changes following DA replacement therapy (levodopa) are correlated with self-reported 'wanting', but not 'liking' (Evans et al., 2006). Brain imaging studies using fMRI confirmed that the expectation (wanting) and receipt (liking) of pleasant tastes activate distinct brain areas (O'Doherty, Deichmann, Critchley, & Dolan, 2002).

The Incentive-Sensitization Theory also proposes that repeated drug use sensitizes, rather than reduces, mesolimbic DA activity. This is supported by animal studies showing enhanced locomotor responses to psychomotor stimulants with repeated injections (Wise & Bozarth, 1987; Robinson & Berridge, 1993) and increases in stereotypy and motor behavioural patterns in chronic drug abusers (Steketee & Kalivas, 2011). Sensitization of mesolimbic DA may also underlie the persistent craving and attentional bias for drugs that

develop with addiction. Neurobiologically, the mesolimbic DA system is altered with repeated drug administration, resulting in increased outflow of DA from pre-synaptic neurons and more DA D1 receptors on the post-synaptic membrane. Structurally, medium spiny neurons in the nucleus accumbens and prefrontal cortex 'grow' more dendritic branches and spines following chronic drug intake, increasing the capacity for DA transmission (Robinson & Kolb, 1997, 1999; Robinson & Berridge, 2000).

A primary characteristic of drug addiction is excessive craving triggered by drug-associated stimuli. For example, in human drug users, imagery, contextual, and social cues previously paired with drug intake can trigger drug use (Weiss et al., 2001; Norberg, Kavanagh, Olivier, & Lyras, 2016). The Incentive-Sensitization Theory explains this maladaptive response through sensitization of DA function: the association between cues predicting drug intake and drug effects are mediated by mesolimbic DA. With sensitization, cues associated with drug use become more salient predictors of drug effects, thereby eliciting craving (K. Berridge & Robinson, 2011).

The Computational Modelling Approach

With the amount of experimental evidence and possible theoretical interpretations of addiction, researchers have turned to computational tools to form and test theories. Many models described addiction as a negative reinforcement process, focusing on analyzing the role of withdraw in addiction (Zhukovsky et al., 2019). But as argued in the previous sections, neither positive nor negative reinforcement provides a full account of addiction. On a behavioral economic level, models of free decision making is often used to study the compulsive behaviour of addiction (Redish, 2004; Morris & Cushman, 2019). On a psychopharmacological level, neuroscientists examine the role of DA activation and transmission in psychiatric disorders (Enrico et al., 2016). The difficulty with these models is that results are generalized across many disorders. Therefore, they fail to identify the unique mechanism that is responsible for the formation of addiction. Levy and Colleagues developed a multiscale model of addiction, integrating cognitive, behavioural and neural psychological factors (Levy, Levy, Barto, & Meyer, 2013) to simulate the development of drinking behavior of a virtual agent. However, because the model consisted of multiple factors, including incentive sensitization, withdraw, rationality, and social influence, among others, it was difficulty to examine each factor individually and more importantly, to make informative claims about the role of each in addiction.

Method

To design an informative and applicable model, we first examined the core proposal of the Incentive-Sensitization Theory (Robinson and Berridge 2000, 2016) which include the following four statements:

'(1) Potentially addictive drugs share the ability to produce long-lasting adaptations in neural systems (i.e., addictive drugs change the brain).

(2) The brain systems that are changed include those normally involved in the process of incentive motivation and reward.

(3) The critical neuroadaptations associated with addiction render brain reward systems hypersensitive ("sensitized") to drugs and drug- associated stimuli.

(4) The brain systems that are sensitized do not mediate the pleasurable or euphoric effects of drugs (drug "liking"), but instead they mediate a subcomponent of reward we have termed incentive salience or "wanting".'

Then, we designed models with structural components that can actualize the processes described in these statements. Particularly, our model is not examining Statement(4),the disassociation of 'liking' and 'wanting'. The reason is three fold: 1) the separation of 'liking' and 'wanting' has been acknowledged, and the notion that mesolimbic DA activity is not responsible for 'liking' is now widely accepted. 2) According to the statement(1), 'wanting', not 'liking' is the essential component of addiction 3) In this model, we describe the neurological changes that are common for all drug types, which is 'wanting'. But different classes of drugs may have different hedonic effects. Therefore, our model will mainly describe the first three statements.

Our goal here is to explore various ways of building a computational implementation of the above theory. That is, we want to examine different possible methods for having biological components that create the addiction process. We used a step by step approach, starting with the smallest processing component and adding more complex features with each model.

We are also constraining our focus to substance addiction alone, excluding behavioural addictions. The goal of the study is to examine the neurological changes and mechanism of addiction. Substances have a more direct impact on neural circuits. Moreover, whether or not is behavioural addiction (such as gambling and pornographic addiction) the same as substances is still under debate (Alavi et al., 2012).

Schematic Description

The architecture of the neural network for simulating incentive sensitization is represented in the following schematic. (presented at CSBBCS conference).

Our model will build up the incentive saliency attributor component of the schematic, which is a main characteristic of the incentive sensitization model of addiction.

Nengo and the Neural Engineering Framework

Since our eventual goal is to have a biologically grounded implementation of Incentive-Sensitization Theory, we decided to implement our models using Nengo, a software package implementing the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003). This forces each component in the model to be something that can be implemented using spiking neurons.

As such, our models consist of five core features of NENGO. (1) Groups of neurons (ensembles) encode numer-

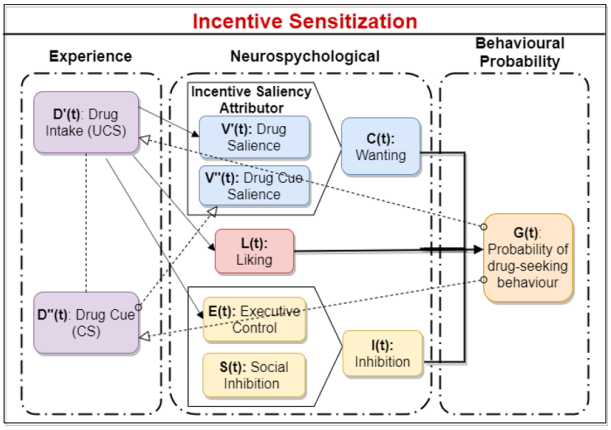


Figure 1: Schematic description of the Incentive-Sensitization Theory.

ical *vectors*, such that different patterns of activity represent different values, using a distributed population code. (2) Connections between ensembles compute functions on those vectors, allowing information to be transforms and transmitted throughout the model. (3) Recurrent connections within an ensemble allow for storage of information over time; technically, this allows the neurons to approximate arbitrary differential equations, allowing the current value represented by the neurons to be a function of both their current input and their value in the recent past. (4) Learning rules allow connection weights to be changed, effectively changing the function that is being computed. (5) Modulation of neuron parameters allows for large-scale changes to a group of parameters of the model, such as making a neuron more sensitive to firing, or scaling how quickly a learning rule changes weights.

We create approximations of the brain's response to drug and drug-associated stimuli by scaling down the duration of drug-use experience and the resulting neurological processes. For example, a drug use episode might take hours, but our models only receive the input of drug intake for several seconds. This shortened time frame allows us to examine the models' behaviour without running full-length simulations while retaining the interpretability of the results.

Models

Model I – Dopamine Activation and Mesolimbic Sensitivity

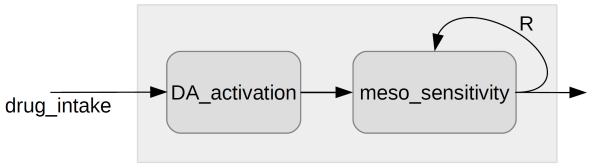


Figure 2: Structure for Model I. Rounded boxes are groups of neurons, arrows are all-to-all connections approximating functions.

Structural Design: Our first model is a direct representation of the top section of the schematic flow: the drug salience attributor. According to the Incentive-Sensitization Theory, the salience value of drugs reflects hyper-reactivity of the mesolimbic system. Stimuli with incentive value, including natural reinforcers, stimulate DA activity in the mesolimbic system. In drug addiction, intensified DA activation also changes the neurological structures responsible for DA activation, leading to a hypersensitized mesolimbic system (Robinson & Kolb, 1999). Therefore, the drug salience attributor should consist of at least two components DA activation and mesolimbic sensitivity. The output of the salience attributor integrates with other mechanisms (described in Model II and III) to form an overall wanting for drugs.

In this model, `drug_intake` is an external stimulus, with a value of 0 or 1. `DA_activation` approximates the value of `drug_intake`. Therefore,`DA_activation` increases when `drug_intake` = 1 and decreases when `drug_intake` = 0. This pattern of DA activation is normally involved in reward processing, as described in Statement(2). Additionally, chronic drug use leads to further neuroadaptations, making the mesolimbic system hypersensitive to drug intake, as described in Statement(3). To store the sensitization of the mesolimbic system, the recurrent `meso_sensitivity` component(defined in the previous section) reflects both the increased baseline synaptic DA transmission reacting to drug intake and the structural changes increasing the capacity of mesolimbic DA activity.

Model Behaviour: According to Statement(1), neuroadaptation with repeated drug use is long-lasting. Therefore, each drug intake should have significant sensitization effects on the mesolimbic system, while the decay of the sensitization during drug absence should be slower. To create simple representations of this mechanism, we computationally manipulated Model I to perform the 4 functions in Table 1.

Table 1: Model I Computational Options

Simulation	R	meso_sensitivity(t) = M(t)
Simulation1	0.9	M(t)
Simulation2	0.9	M(t)+0.1 if DA_activation=1 M(t)-0.01 if DA_activation= 0
Simulation3	0.9	M(t)+0.15 if DA_activation=1 M(t)-0.01 if DA_activation= 0
Simulation4	0.9	M(t)+0.2 if DA_activation=1 M(t)-0.01 if DA_activation= 0

In Simulation1, `meso_sensitivity` has a recurrence value of 0.9, storing 90% of the neuroadaptations made at the previous time point. Simulation2, Simulation3 and Simulation4 added a non-linear function, where `meso_sensitivity` is increased by a certain amount with spikes in DA activation and is decreased when `DA_activation` is minimal.

To examine Model I, we fed in 0.6 seconds of `drug_intake` = 1 following 0.4 seconds of drug absence.

This relatively long duration of drug administration (0.6s out of every 1s) provides a clear visual demonstration of its effect on mesolimbic DA sensitivity. `Meso_sensitivity` is recorded as the output. With six repetitions of `drug_intake`, the four simulations are compared as shown in Figure 3.

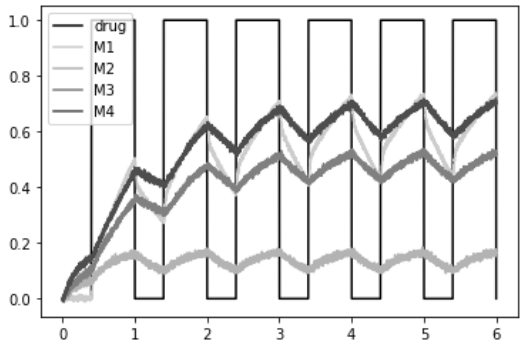


Figure 3: meso_sensitivity output of 4 variants of Model I

Model Evaluation: Simluation1 showed a significant overall growth in `meso_sensitivity` with drug intake repetition. Before plateauing, `meso_sensitivity` in Simulation1 had a greater increase during drug intake than its decrease during drug absence. Simulation2 and Simulation3 had lower overall increases of mesolimbic sensitivity compared to the other two simulations. In Simulation4, the amount of increase in `meso_sensitivity` with drug intake was 20 times greater than decreases in between drug intakes. Therefore, if the mesolimbic system achieves sensitization by implementing the nonlinear function in Model I, it must be 20 times more efficient in developing sensitization than to decay the sensitized information. Comparing Simulation1 to Simulation4, although they reached similar levels of mesolimbic sensitivity with repeated drug intake, Simulation1 had a greater spike in mesolimbic sensitivity in response to each drug intake. This characteristic of Simulation1 correlates with the hyperactivity feature of a sensitized mesolimbic system described in the Incentive-Sensitization Theory. However, it can also be argued that Simulation4 reached a higher baseline of wanting during drug absence, which also coincides with the Incentive-Sensitization Theory.

All four simulations in Model I reached a maximum level of `meso_sensitivity` within 5 representations of drug intake. In contrast, pathological drug use is often characterized by a ramping up of craving for drug (i.e., wanting) over a longer period. Thus, with the current structural design, Model I failed to describe the pattern of wanting in addiction. Therefore, other structural designs are required to achieve a continual increase in wanting with repeated drug use.

Model II - Drug Cue Salience

Structural Design: The sensitization track, from `drug_intake` to `DA_activation` to `meso_sensitivity` is the same as Model II. On top of that, we added drug-associated cue processing. Drug-associated stimuli are

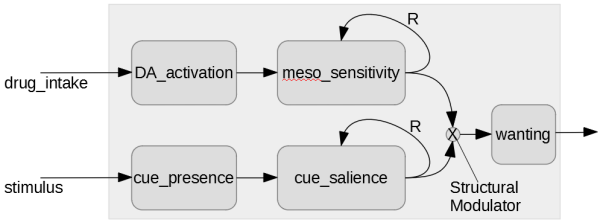


Figure 4: Structure of Model II. Circled X is a multiplicative modulation of connection strengths.

conditioned stimuli, so their representation leads to neurological representation of drug-associated effects. With repeated exposure to the drug cue, salience attributed to the drug cue will accumulate, according to Statement(2) of the Incentive-Sensitization Theory. Overall, drug intake should increase the overall baseline of wanting for drugs, while the drug cue should elicit acute increases in wanting the drug.

The structural modulator implements the two primary ways that mesolimbic sensitization and associative learning can affect each other. The learned cue salience may directly impact the mesolimbic system, triggering intensified wanting. Alternatively, the sensitized mesolimbic system may amplify cue salience, eliciting exaggerated wanting. These are done as affine transformations from 0-1 to 1-10.

Model Behaviour: To examine the model, we feed in `drug_intake`=1 every 0.65 seconds with a drug presence of 0.1 seconds, and a stimulus every 0.9 seconds, for 0.1 seconds. This set up allows the independent presence of drug and cue, as well as a combined presence of both drug intake and cue at t=3 and t=6. The value of `wanting` in Simulation1 and Simulation2 are compared in Figure 5.

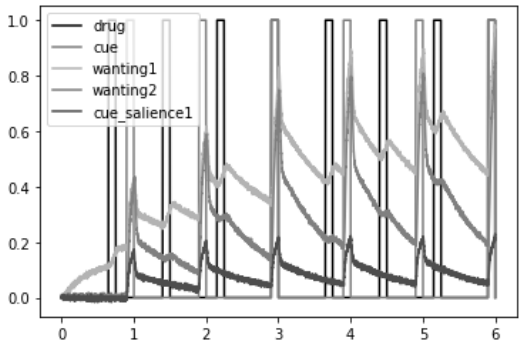


Figure 5: Wanting in simulation1 and 2 are presented in green and red respectively. cue_salience1 is the value of cue_salience in simulation 1.

Model Evaluation: Both simulations had an accumulated increase of wanting at t=6 compared to baseline at t=0. Compared to Model I, the two simulations in Model II also had a continual increase, extending beyond the plateau in Model I. However, Simulation1 had an overall steeper increase in wanting than Simulation2. While drug cue triggered similar spikes of wanting in both simulations, drug intake resulted

in a greater increase in wanting in Simulation1 than in Simulation2. Importantly, `wanting` failed to respond to the first presentation of `drug_intake`. Therefore, the computational design in Simulation2 did not generate enough increase in baseline wanting. Thus, Simulation1 is a more realistic representation of addiction formation.

`Cue_salience` in Simulation1 showed a slight increase with repeated drug cue presentation. Nonetheless, with the computational design in the `structural_mod` component, `wanting` achieved continual growth in Simulation1, compared reaching an early plateau in Model I. So far, we have adjusted the degree of change in mesolimbic sensitivity and cue salience, while leaving the two processes independent of each other. Another way to create an extraordinary level of wanting is to have sensitization and associative learning processes dependent on each other. The Incentive-Sensitization Theory of addiction emphasizes the abnormal amount of additional bias given to drug-associated cues. Therefore, it is plausible that the sensitized mesolimbic system can intensify the associative learning drug cues, triggering more craving for drugs.

Model III - Intensified Associative Learning

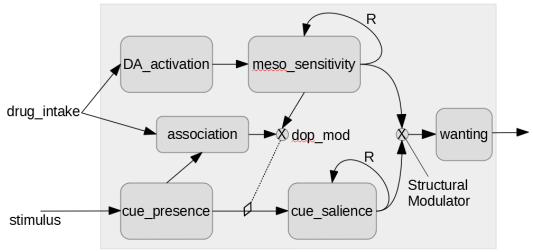


Figure 6: Structure of Model III. Dotted line is a learning signal, adjusting the strengths of the connection weights to which it is connected.

Structural Design: The main feature of this model is that mesolimbic sensitivity can accelerate the formation of drug cue salience. To implement this feature, the `association` component tracks the presentation of `drug_intake` and `cue_presence`, computing the absolute distance of the two values. The stronger the `association`, the more `cue_salience` should increase. Then, `dop_mod` scales up `association` based on `meso_sensitivity`. The integration of `association` and `meso_sensitivity` then contributes to the development of `cue_salience`. Therefore, when drug intake and stimulus are presented close together (aka, there is an association), a sensitized mesolimbic system should intensify the power of association, leading to a faster increase in cue salience.

Model Behaviour: Our primary purpose in this simulation is to test the new `association` and `dop_mod` components. To reduce uncertainty in the other processes in the model, we used the linear option for `meso_sensitivity`, with a recurrence value of 0.95. We fed the model with `drug_intake`=1 every 0.65 seconds with a drug presence of 0.1 seconds, and stimulus every 0.9 seconds, for 0.1 seconds. With ten repetitions of the inputs, results are shown as below:

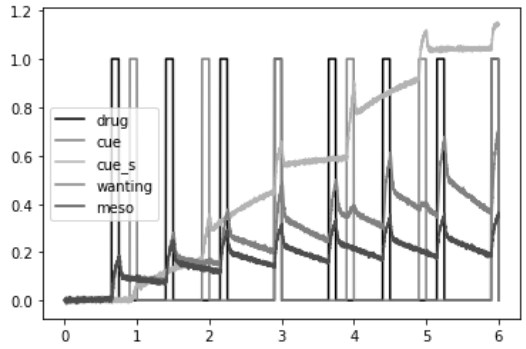


Figure 7: Cue_salience is presented as cue_s in green. Wanting is in red, and meso_sensitivity is meso in purple.

Model Evaluation: Model III simulation generated overall increases in mesolimbic sensitivity, wanting and cue salience. Cue salience in Model III increased with a significantly steeper slope than the `cue_salience` curve in Model II Simluation1 (Figure 5). In Model II, `cue_salience` did not increase as drug intake, and drug cue built an association. In contrast, the new structural design in Model III helped the simulation achieve an overall growth in `cue_salience`. Moreover, the spike in `cue_salience` following drug cue presentation also increased as drug cue presentation repeated. The success of associative learning is also evident in the pattern of wanting. Importantly, at the first two cue presentation, wanting showed minimal increases. At t = 3, drug intake and drug cue occurred together, forming a strong association. Subsequently, at t=4 and t=5, cue presentation elicited a noticeable increase in wanting. In other words, in accordance with the Incentive-Sensitization Theory, Model III simulation demonstrated sensitization in mesolimbic reactivity, drug cue related response, and overall wanting for drugs.

Discussion

Overall, our study took a structuralist computational approach, rebuilding the neurological process of addiction according to the blueprint provided by the Incentive-Sensitization Theory of addiction. Here, we discuss the implications of our model simulations in relation to the first three statements of the Incentive-Sensitization Theory, the limitations to our models, and future directions for studies on drug addiction. Implementing Statement(1), the long-lasting sensitization: In this paper, we explored both linear and non-linear options of implementing the long-lasting effect of sensitization. Simulations in Model I demonstrated that both linear and non-linear functions are computationally plausible to describe mesolimbic sensitization.

The incentive sensitization did not propose a computational guideline that specifies the degree of long-lasting effect. To determine the exact value of recurrence strength in the linear function, or the variables in the non-linear func-

tion, we need a realistic data set. Ideally, this would contain the speed of sensation formation and decay, represented by quantifiable units (e.g., the duration of time, or the number of drug use repetitions required to produce sensitization). Then we can perform data fitting and select the model with the best fit.

Implementing Statement(2), brain areas responsible for motivation: While we can pinpoint the mesolimbic system as the centre for processing rewards, it is difficult to determine the brain area responsible for cue salience attribution. We have an adaptive bias towards natural rewards such as food. But according to the Incentive-Sensitization Theory, chronic drug use will produce an abnormal attentional bias towards drug-associated cues (Robbinson and Berridge, 2003). Therefore, other than sensitization in the mesolimbic system, there must be a neural activation pattern that is dedicated to directing sensitization towards drug-associated cues. When implementing `cue_salience` in our models, we did not specify the neurological areas corresponding to this computational component. The `structural_modulator` in Model II and III was also not implemented as a neurological component. This is because current literature of incentive sensitization does not account for the neurological mechanism of how drug cue salience triggers wanting in incentive sensitization.

Implementing Statement(3), drug salience and cue salience associative learning: Incorporating the cue association process, Model II generated more realistic simulations compared to Model I. Model II extended the system's capacity for growth of wanting, allowing for the continual increase in wanting beyond the plateaus produced in Model I. Furthermore, Model I simulated two possible mechanisms of combining mesolimbic sensitization and drug cue associative learning. The results supported the hypothesis that intensified drug cue salience affects the sensitized mesolimbic system, triggering spikes in wanting rather than the other way around. Nonetheless, there are limitations to the current model. Crucially, neither Model II nor III included a complete process of the development of drug cue associative learning. For example, our simulations registered a successful pairing of drug and drug cue only when `drug_intake` and stimulus are presented simultaneously. In reality, associative strength is strongest when the conditioned stimulus is presented slightly after the unconditioned stimulus. Future studies can utilize mature computational models of associative learning to complete the model further.

Models in this paper are shortened approximations of real-life neurological processes. The assumption is that if the models' time scale is scaled up to match actual drug-use experiences, the qualitative brain changes should be the same. However, this assumption requires validation with human data and more extended simulations that run for weeks, if not months. Extensive simulations might also reveal a faster rate of Incentive-Sensitization formation and a slower rate of its recovery. This is because, in our current models, the ratio of drug intake duration to drug absence duration is significantly higher than in reality. For instance, Model I Simulation 4 had drug intakes of 0.6 seconds and drug absences of 0.4 seconds – the length of drug absence is 2/3 of drug intake. In reality, drug absence can last days, weeks, even months. Drug users with longer abstinence in between drug repetitions still present a high level of wanting for drugs. Moreover, the duration of drug intake is usually shorter than drug absence. Importantly, the neuropharmacological effects of addictive drugs can occur within minutes after drug intake. This means that the Incentive-Sensitization of the mesolimbic DA system can form within minutes of drug use and remain robust after drug absence periods longer than in the models. In Model I Simulation 4, the recurrent strength of the mesolimbic DA system is set to 0.9. The increase of mesolimbic sensitivity during drug intake is 20 times larger than its decrease during drug absence. Given the analysis above, the computational difference between mesolimbic sensitivity increase and decrease is likely to be more significant with more realistic simulations. Overall, it is likely that the strength of Incentive-Sensitization as a result of addictive drug use is more robust than the models present.

In sum, our paper informs future studies in that 1) the rate of mesolimbic sensitization can be determined with data fitting from clinical studies, 2) the process of associative learning indispensable to drug addiction formation, 3) the Incentive-Sensitization Theory needs to identifying the neurological area responsible for storing associated salience to drug cues, and 4) the methodology of computationally structuring virtual neural circuits can be very useful for examining theories of neurological mechanisms and simulating those processes.

References

Alavi, S. S., Ferdosi, M., Jannatifard, F., Eslami, M., Alaghemandan, H., & Setare, M. (2012). Behavioral addiction versus substance addiction: Correspondence of psychiatric and psychological views. *Int J Prev Med*, *3*(4), 290–294.

Berridge, K., & Robinson, T. (2011). Drug addiction as incentive sensitization. *Addiction and responsibility*, 21-54.

Berridge, K. C. (2007). The debate over dopamine's role in reward: the case for incentive salience. *Psychopharmacology*, *191*(3), 391–431.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems* [book]. Cambridge, MA: MIT Press.

Enrico, P., Migliore, M., Spiga, S., Mulas, G., Caboni, F., & Diana, M. (2016). Morphofunctional alterations in ventral tegmental area dopamine neurons in acute and prolonged opiates withdrawal. a computational perspective. *Neuroscience*, *322*, 195–207.

Evans, A. H., Pavese, N., Lawrence, A. D., Tai, Y. F., Appel, S., Doder, M., ... Piccini, P. (2006). Compulsive drug use linked to sensitized ventral striatal dopamine transmission. *Annals of Neurology*, *59*(5), 852–858.

Farooqi, I. S., Bullmore, E., Keogh, J., Gillard, J., O'Rahilly, S., & Fletcher, P. C. (2007). Leptin regulates striatal regions and human eating behavior. *Science*, *317*(5843), 1355.

Koob, G. F. (2020). Neurobiology of opioid addiction: Opponent process, hyperkatifeia, and negative reinforcement. *Biological Psychiatry*, *87*(1), 44–53.

Koob, G. F., & Le Moal, M. (1997). Drug abuse: hedonic homeostatic dysregulation. *Science*, *278*(5335), 52–58.

Koob, G. F., & Volkow, N. D. (2016). Neurobiology of addiction: a neurocircuitry analysis. *The Lancet Psychiatry*, *3*(8), 760–773.

Levy, Y. Z., Levy, D. J., Barto, A. G., & Meyer, J. S. (2013). A computational hypothesis for allostasis: Delineation of substance dependence, conventional therapies, and alternative treatments. *Front. Psychiatry*, *4*.

Morris, A., & Cushman, F. (2019). Model-free RL or action sequences? *Front. Psychol.*, *10*.

Norberg, M. M., Kavanagh, D. J., Olivier, J., & Lyras, S. (2016). Craving cannabis: a meta-analysis of self-report and psychophysiological cue-reactivity studies. *Addiction*, *111*(11), 1923–1934.

O'Doherty, J. P., Deichmann, R., Critchley, H. D., & Dolan, R. J. (2002). Neural responses during anticipation of a primary taste reward. *Neuron*, *33*(5), 815–826.

Pierce, R. C., & Kumaresan, V. (2006). The mesolimbic dopamine system: the final common pathway for the reinforcing effect of drugs of abuse? *Neurosci Biobehav Rev*, *30*(2), 215–238.

Pool, E., Sennwald, V., Delplanque, S., Brosch, T., & Sander, D. (2016). Measuring wanting and liking from animals to humans: A systematic review. *Neurosci Biobehav Rev*, *63*, 124–142.

Redish, A. D. (2004). Addiction as a computational process gone awry. *Science*, *306*(5703), 1944–1947.

Robinson, T. E., & Berridge, K. C. (1993). The neural basis of drug craving: an incentive-sensitization theory of addiction. *Brain Res. Brain Res. Rev.*, *18*(3), 247–291.

Robinson, T. E., & Berridge, K. C. (2000). The psychology and neurobiology of addiction: an incentive-sensitization view. *Addiction*, *95 Suppl 2*, S91–117.

Robinson, T. E., & Kolb, B. (1997). Persistent structural modifications in nucleus accumbens and prefrontal cortex neurons produced by previous experience with amphetamine. *J. Neurosci.*, *17*(21), 8491–8497.

Robinson, T. E., & Kolb, B. (1999). Alterations in the morphology of dendrites and dendritic spines in the nucleus accumbens and prefrontal cortex following repeated treatment with amphetamine or cocaine. *Eur. J. Neurosci.*, *11*(5), 1598–1604.

Shaham, Y., Rajabi, H., & Stewart, J. (1996). Relapse to heroin-seeking in rats under opioid maintenance: the effects of stress, heroin priming, and withdrawal. *J. Neurosci.*, *16*(5), 1957–1963.

Steketee, J. D., & Kalivas, P. W. (2011). Drug wanting: behavioral sensitization and relapse to drug-seeking behavior. *Pharmacol. Rev.*, *63*(2), 348–365.

Weiss, F., Ciccocioppo, R., Parsons, L. H., Katner, S., Liu, X., Zorrilla, E. P., ... Richter, R. R. (2001). Compulsive drug-seeking behavior and relapse. neuroadaptation, stress, and conditioning factors. *Ann. N. Y. Acad. Sci.*, *937*, 1–26.

Wise, R. A. (1980). Action of drugs of abuse on brain reward systems. *Pharm. Biochem. Behav.*, *13 Suppl 1*, 213–223.

Wise, R. A., & Bozarth, M. A. (1987). A psychomotor stimulant theory of addiction. *Psychol Rev*, *94*(4), 469–492.

Zhukovsky, P., Puaud, M., Jupp, B., Sala-Bayo, J., Alsiö, J., Xia, J., ... Dalley, J. W. (2019). Withdrawal from escalated cocaine self-administration impairs reversal learning by disrupting the effects of negative feedback on reward exploitation: a behavioral and computational analysis. *Neuropsychopharmacology*, *44*(13), 2163–2173.

A grammatically robust cognitive model of English and Korean sentence processing

Stephen Jones (s.m.jones@rug.nl)
Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence,
University of Groningen, 9700AB Groningen, Netherlands

Abstract

This paper presents a novel approach to the cognitive modelling of human sentence processing in ACT-R. The model assumes a cognitive distinction between cross-linguistic knowledge of the overall possibilities for combining elements of language structure, represented in procedural memory, and language-specific knowledge of the combinatorial constraints on structure-building, which are stored as part of the lexicon in declarative memory. Sentence structure is built incrementally using an extension of an established, computationally robust grammar theory, Lexical Functional Grammar (Bresnan, 1982). Using a single set of productions, together with a dual lexicon representing grammar fragments of English and Korean, the model is able to parse complex sentences in both languages, constructing syntactic representations that match human judgements. The model reproduces garden path phenomena reported by English and Korean native speakers, and introduces a cross-linguistic treatment of prosodic breaks to avoid garden-paths during processing. Limitations to the model are discussed, as well as questions that are currently under investigation.

Keywords: ACT-R; Lexical Functional Grammar; sentence processing; parsing; syntactic structure; English; Korean.

Introduction

ACT-R has been used to construct models of human language processing for more than twenty years. The aims of individual models have varied, including exploring the nature of memory for language processing (Anderson, Budiu, & Reder, 2001; Budiu & Anderson, 2004), and the extraction of meaning from text input in real time (e.g Ball, 2011). The preeminent model (R. L. Lewis & Vasishth, 2005, henceforth LV05) demonstrated that the retrieval of memory chunks representing syntactic structure could replicate the processing time-courses for English sentences with differing levels of complexity. The mathematical underpinning of LV05 has been used to extend the model to other languages without the need to create full structural representations; this subsequent work has raised questions about some of the assumptions in the original model (e.g. Jäger, Engelmann, & Vasishth, 2015).

All of the models were designed to address specific research questions, but these design choices reduce the models' generalisability from linguistic and cognitive modelling perspectives. With regard to cognitive modelling, all of the models assume extra working memory capacity in the form of overt or hidden buffers that are each linked to one a set of assumed phrasal categories. Thus in principle the size of additional buffer capacity is limited only by the chosen grammar theory rather than cognitive considerations. Left-corner parsing is a common assumption, but this requires either a stack or additional memory capacity in place of a stack. Assuming some additional capacity may be reasonable, but it is still an open question as to how capacity can be added parsimoniously.

The extent to which speakers' acquired knowledge of grammar affects real-time processing is very much live in psycholinguistics (e.g. S. Lewis & Phillips, 2015). Even theories such as Good Enough Processing (Ferreira, Bailey, & Ferraro, 2002) require specificity in their descriptions of syntactic and semantic processing. For cognitive models to address this question productively, their representations of syntax, and of the syntax-semantics interface, need to be theoretically grounded and generalisable.

From a linguistic perspective, existing models make questionable assumptions about structural representations and the relationship between syntax and meaning. Previous models have generated binary branching tree structures based on versions of X-bar theory (Jackendoff, 1977). However, non-configurational languages such as Wambaya (Nordlinger, 1998) provide scant evidence of a binary-branching structure. Models that rely on binary trees thus restrict themselves to phenomena from a subset of languages, rather than considering the general human language faculty. The relationship between syntactic structural position and meaning is either stipulated or falls outside the scope of the model, rather than being grounded in linguistic theory.

The pervasive ambiguity of language is a challenge for modellers, and the stimuli chosen for psycholinguistic experiments often complicate the task of modelling. Ambiguity is typically addressed by working with a fragment of a grammar, and by assuming that the model has knowledge unavailable to native speakers, e.g. predicting the structure that will follow a particular sentence opening, or distinguishing in advance between particular types of clause. Models developed in this way sidestep some of ACT-R's architectural constraints, but restrict their generalisability to other phenomena or languages, and weaken their link to linguistic theory.

In the light of these deficiencies, there is still a need for models of language processing that are based on robust theories of grammar beyond constituent structure, and which seek to generalise a processing model across different languages.

Grammar formalism

Lexical Functional Grammar (LFG) is a modular, constraint-based theory that permits accounts of language phenomena across syntax, semantics, information structure, discourse, and sentence prosody. Syntactic constraints are distributed across two levels of representation: c-structure, which represents surface constituent structure governed by language-specific phrase-structure rules; and f-structure, a universal representation of the functional relationships between meaning-bearing elements of the sentence. The sentence-based theory has been expanded to an incremental theory of sentence growth, in which c-structure and f-structure constraints interact to restrict the possibilities for new information to be added to an emerging structure. F-structure is not only language-independent, but is also the base from which semantic and discourse representations are projected. Thus the representation of syntactic structure in the model is based on f-structure.

The model

The model assumes that language-specific grammatical knowledge is encoded in the lexicon and stored in declarative memory. Additional working memory capacity is assumed in the form of three additional buffers, each loosely associated with events (verbs), things (nouns) or qualities (adjectives/prepositions). Incomplete phrases are maintained in working memory and a multibuffer (Salvucci & Taatgen, 2008) allows the processing of embedded clauses. Each new word is attached into structure before the next is read. The combinatorial possibilities for attachment are encoded in procedural memory, where a single production set is used for both English and Korean sentences.

The generated sentence structure is a graph composed of chunks that represent f-structure. Each chunk other than the root has one mother chunk, and a chunk's relationship to its mother is specified as a grammatical function. The structure is not binary-branching: because of this and ACT-R's assumptions on the growth of information held in a chunk, it was not possible to follow LV05 in modelling retrieval effects on processing time-courses. I return to this in the Discussion.

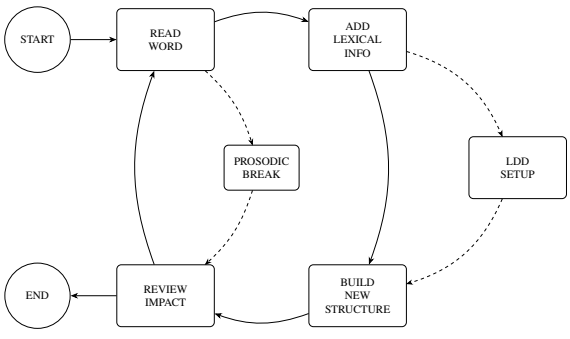


Figure 1: The model parsing cycle

Figure 1 shows the parsing cycle assumed in the model. The model assumes full incremental processing, with only one structural representation maintained. Each word is integrated into the structure before the next is processed. The representation of grammatical functions is based on decomposition into features. An attachment site is chosen by comparing the feature set of the word to be processed is with the unfilled grammatical functions of structural chunks in the buffers. This has two consequences that reflect human behaviour in processing. One is that attachment is context dependent: identical strings are processed differently depending on prior content. The second is that linguistic ambiguities or alternations can be expressed by means of partial feature specifications, allowing flexible combination of words but still generating fully specified output.

The model further assumes that prosodic breaks act to signal the end of a syntactic phrase and force a structural chunk to be cleared from a buffer. This allows the modelling of prosodically-modulated garden path effects.

Results

Results are presented here to illustrate four properties of the model: the ability to analyse complex sentences in English and Korean; the role of feature underspecification in processing structural alternations; context-dependent parsing of identical strings; and the representation of prosodic breaks in guiding the path of structure-building. All figures showing structural representations are given at the end of the paper.

Comparing outputs between languages The model provides an equivalent representation of sentences that have the same meaning in English and Korean, despite the significant variation in constituent structure between the two languages. Korean constituents are strongly head-final, with no fixed ordering of arguments to a verb. Relative clauses precede their head and are marked morphologically at the end of the clause. English has a fundamental SVO word order, with relative clauses appearing after their heads and optionally marked by a complementiser.

Sentences (1) and (2), taken from Kwon, Gordon, Lee, Kluender, and Polinsky (2010), carry broadly equivalent meanings in the two languages (the English sentence lacks an adverb), but different word orders, as can be seen from the gloss in (2). In example sentences, relative clauses are indicated by square brackets.

(1) The conductor [who the famous vocalist invited to the festival] insulted the senator.

(2) [*yumyenghan sengakkaka chwukceney chotayhan*]
[famous vocalist.SUBJ festival.LOC invited]
cihwuycaka uywonul kongkongyenhi
conductor.SUBJ senator.OBJ publicly
moyokhayssta
insulted

"The conductor [who the famous vocalist invited to the festival] publicly insulted the senator."

Figures 2 and 3 show the outputs for the English and Korean sentences respectively. The chunk indices reflect the different word order, and the difference between the chunks 'pro' and 'PRO' reflects the presence or absence of a lexical complementiser in a relative clause[1].

Underspecification The model uses underspecification of grammatical functions to handle lexical ambiguity, without assuming knowledge that is not available to human subjects. The English verb *give* freely alternates between a form that takes a subject, plus two noun phrases as objects, and a form in which the third argument is a prepositional phrase. The choice of argument structure determines the distribution of semantic roles, and until the third argument is processed, its grammatical function, and the resulting semantic roles, are ambiguous. The sentences in (3) show alternate argument structures for the same event. In each case the second and third arguments of *give* are shown in *italics* and **bold** respsectively.

(3) a. The conductor gave *the senator* **a gift**.
b. The conductor gave *a gift* **to the senator**.

In sentence (3a), the grammatical functions of *give* are ⟨[SUBJ, OBJ, OBJ_θ]⟩, with OBJ providing the semantic Goal. However in (3b), the grammatical functions are ⟨[SUBJ, OBJ, OBL_θ]⟩ and OBJ provides the semantic Theme.

Table 1: Underspecification and argument alternation

	−R	+R
−O		OBL_θ
+O	OBJ	OBJ_θ

give	SUBJ	*unrestricted*	
	OBJ	+O	−R
	GF_3		+R
gift		+O	
senator		+O	
to		−O	

Table 1 shows the feature system used by the model: the left-hand table gives the full specification for the different grammatical functions using the features ±R 'restricted' and ±O 'object'. The specification on the right shows the features associated with grammatical functions and with words that might fulfil a grammatical function. The third grammatical function of *give* is underspecified as +R, meaning that it could be either OBL_θ or OBJ_θ. When the word *gift* is encountered in (3a), its feature +O combines with +R to fully specify the grammatical function as [+O +R], giving OBJ_θ. Conversely when *to* is encountered in (3b), the resulting feature set is [−O +R], giving the grammatical function OBL_θ.

Context-dependent parsing The parser successfully uses context to distinguish between different meanings of identical strings. Where a required argument of a verb is missing (a governed grammatical function), the parser preferentially attaches there, rather than provide an adjunct. Consider the English string *the boy the dog bit*, which has two sequential noun phrases *the boy* and *the dog* followed by a verb *bit*. Figure 4 shows that in subject position, *the dog bit* is interpreted as a reduced relative clause. The verb of the main sentence is not yet known, and so there are no required grammatical functions which can be assigned to the second noun phrase *the dog*. Thus *the dog bit* attaches as an adjunct to *the boy*.

In Figure 5, the context contains the main verb *give*, which requires two subsequent arguments. Thus *the dog* is attached as the second object of *give*. This results in an incoherent structure after *bit* is processed, representing the garden path effect observed in human subjects.

Prosodic disambiguation

Sentence (4) produces a garden path effect for Korean speakers because it initially appears that the relative clause includes the first two words of the sentence, *yumyenghan cihwuycaka*, 'famous conductor'.

(4) *yumyenghan cihwuycaka* [*sengakkalul*
famous conductor.SUBJ [vocalist.SUBJ
chwukceney chotayhan] *uywonul* *kongkongyenhi*
festival.LOC invited] senator.OBJ publicly
moyokhayssta
insulted
"The famous conductor publicly insulted the senator [who invited the vocalist to the festival]."

Figure 6 shows the structure generated by the model for this sentence: although the representation is structurally grammatical, it is semantically incoherent and requires reanalysis to derive the intended meaning.

If a prosodic break is inserted between the second and third words, the garden path effect disappears and the desired meaning is easily accessible to native speakers. The model simulates this by clearing the active buffer, indicating the right edge of a phrase. The output of the model with prosodic support is shown in Figure 7.

Discussion

While previously published ACT-R models of sentence processing have been successful in developing accounts of the role of memory in human sentence processing, there are aspects of them that are theoretically problematic, including unconstrained additional working memory capacity, assumptions of knowledge unavailable to a human subject, and grammar theories and structural representations that are non-standard and not generalisable.

The model presented addresses many of the criticisms of previous models. Its three additional buffers are a limited amount of additional working memory capacity. It can process ambiguous structures without requiring specificity unavailable in human language, and it is based on standard assumptions of phrase structure without relying on distinct representations for relative vs. main clauses (cf. LV05). It can

[1] In Korean *yumyenghan* 'famous' is a verb and morphosyntactically forms a relative clause.

derive similar structures from two typologically different languages, thus separating the processing of a specific language from general cognitive capability. It is also in principle extensible to other languages with different degrees of configurationality. However, it lacks the ability to reproduce processing time-courses and does not use the structural retrieval mechanism successfully proposed by LV05.

There are two main reasons for this. The first arises from interactions between LFG and ACT-R. Functional structure in LFG is not binary branching: all of the grammatical functions associated with a particular word are contained in a single structural unit. Thus any retrieval-based mechanism needs to make multiple retrievals to build a complete structure, adding new information to a chunk at each retrieval. However, adding information to a retrieved and copied chunk means that on release back into declarative memory, it cannot merge with the chunk from which it was copied. There is a proliferation of the chunks containing the information of one f-structure, each representing a different stage of the emerging structure, and these chunks interfere with subsequent retrievals, causing non-human-like errors.

The second reason is more general, arising from the ambiguity inherent in language. It is often unclear whether or not a word attaches into an existing phrase or starts a new phrase. The model presented manages this by holding incomplete phrases in a buffer. Thus if attachment to the chunk in the buffer is possible, the new word attaches there, and if not, or if the buffer is empty, a new structural chunk is created to allow attachment. In a retrieval-based model, chunks would not be maintained in buffers, and so it will only become apparent after a retrieval failure that a new chunk must be created. In ACT-R, this serial process will result in time-course effects that are not seen in human processing data.

Research in progress

Both of the barriers mentioned require changes to architectural assumptions in ACT-R. Work is in progress to develop a language-specific module based on amended assumptions in two areas.

The first area relates to the behaviour of language structure chunks on release into declarative memory. The aim is to allow chunk merger not only for identical chunks, but also where a chunk has added information monotonically compared to the chunk from which it was copied. This addresses the problem of chunk proliferation and the consequent non-human errors in structural analysis. The second area of work is to allow a buffer request *retrieve-or-create*, that either retrieves a chunk against a specification, or creates a new chunk in the case of retrieval failure, without requiring two separate productions. This increases the capacity of models to process ambiguous structures without assuming prescient knowledge, without adding unnecessary processing steps that do not reflect human data.

Once complete, the model will be in a position to generate time-courses that are testable against human data. However, it was not possible to include outputs in this paper.

Conclusion

Cognitive modelling has a role to play in addressing a live question in psycholinguistics: the extent to which grammatical knowledge is accessed on-line during language processing. To engage effectively in the debate, models must be both grammatically and cognitively robust, and generalisable beyond specific phenomena or specific languages. The model presented here is grammatically robust, and is not restricted to a single language. It has cognitively plausible elements in that it does not include a stack, and it assumes only limited additional cognitive capacity. It is not yet able to model time-courses, which requires the development and testing of a model with different architectural assumptions to core ACT-R. However, it offers a step towards a model of language processing that addresses the deficiencies of previous work.

References

Anderson, J. R., Budiu, R., & Reder, L. M. (2001). A theory of sentence memory as part of a general theory of memory. *Journal of Memory and Language*, *45*(3), 337–367.

Ball, J. T. (2011). A pseudo-deterministic model of human language processing. In *Proceedings of the cognitive science society* (Vol. 33, pp. 495–500).

Bresnan, J. (1982). *The mental representation of grammatical relations*. Cambridge, Mass: MIT Press.

Budiu, R., & Anderson, J. R. (2004). Interpretation-based processing: a unified theory of semantic sentence comprehension. *Cognitive Science*, *28*, 1-44.

Ferreira, F., Bailey, K. G., & Ferraro, V. (2002). Good-enough representations in language comprehension. *Current Directions in Psychological Science*, 11-15.

Jackendoff, R. S. (1977). *$\bar{X}$ syntax: A study of phrase structure*. Cambridge, MA.: MIT Press.

Jäger, L. A., Engelmann, F., & Vasishth, S. (2015). Retrieval interference in reflexive processing: experimental evidence from Mandarin, and computational modeling. *Frontiers in Psychology*, *6*(617), 1-24.

Kwon, N., Gordon, P. C., Lee, Y., Kluender, R., & Polinsky, M. (2010). Cognitive and linguistic factors affective subject/object asymmetry: An eye-tracking study of prenominal relative clauses in Korean. *Language*, *86*(3), 546-582.

Lewis, R. L., & Vasishth, S. (2005). An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, *29*, 375-419.

Lewis, S., & Phillips, C. (2015). Aligning grammatical theories and language processing models. *Journal of Psycholinguistic Research*, *44*, 27-46.

Nordlinger, R. (1998). *Constructive case: Evidence from Australian languages*. Stanford, CA.: CSLI Publications.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review*, *115*(1), 101-130.

Figures

Complex sentences in English and Korean

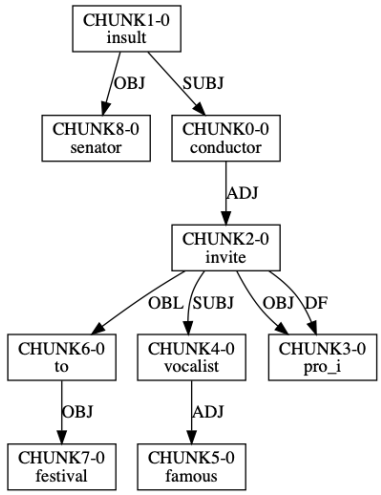


Figure 2: Output from processing sentence (1)

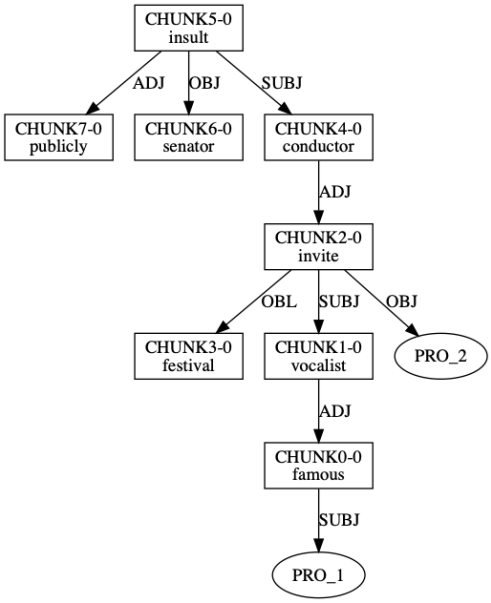


Figure 3: Output from processing sentence (2)

Context-dependent parsing

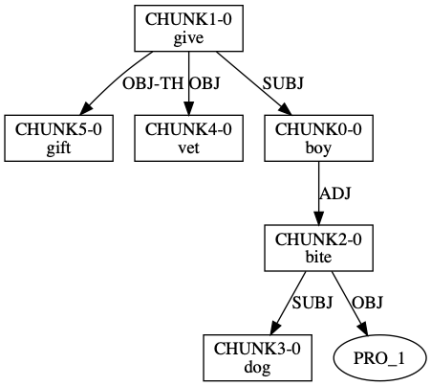


Figure 4: *The boy the dog bit* gave the vet a gift.

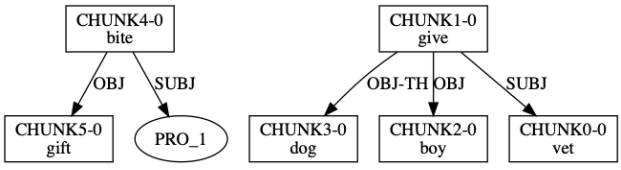


Figure 5: The vet gave *the boy the dog bit* a gift.

Prosodic disambiguation

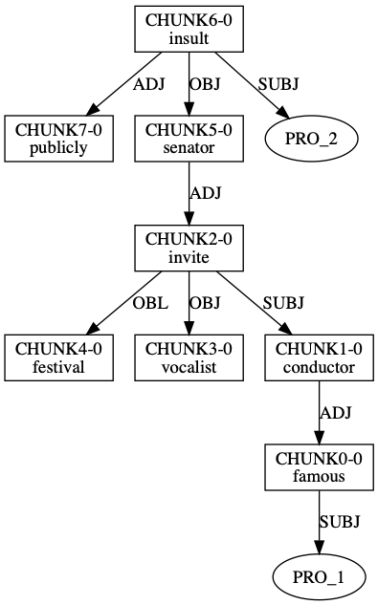


Figure 6: Output from processing sentence (4) *without* prosodic support

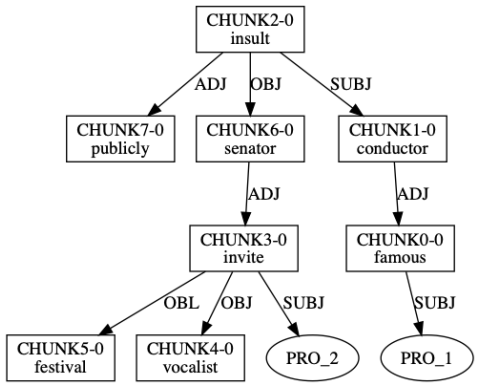


Figure 7: Output from processing sentence (4) *with* prosodic support

Decoding the Mental States of Focus and Distraction in a Real Life Setting of Tibetan Monastic Deabtes Using EEG and Machine Learning

Pallavi Kaushik (pkaushik@cs.iitr.ac.in)
Bernoulli Institute, University of Groningen,
Bernoulliborg, Nijenborgh 9, Groningen, the Netherlands
And
Department of Computer Science and Engineering, Indian Institute of Technology Roorkee
Roorkee, India

Marieke van Vugt (m.k.van.vugt@rug.nl)
Bernoulli Institute, University of Groningen
Bernoulliborg, Nijenborgh 9, Groningen, the Netherlands

Partha Pratim Roy (proy.fcs@iitr.ac.in)
Department of Computer Science and Engineering, Indian Institute of Technology Roorkee
Roorkee, India

Keywords: EEG; Machine Learning; Focus; Distraction; Real Life Scenario

Introduction

EEG data are usually collected in the context of a contrived laboratory setting, which makes it challenging to make inferences about the real world. Here we are presenting machine learning methods that can be used to parse more complex and ecologically-valid settings for collecting EEG data. We will focus on the real-life situation of the monastic debate engaged in by Tibetan monks. The mental process we will examine is that of focus and distraction, for which neural correlates are well-known and fairly robust, typically consisting of posterior alpha oscillations [1].

Why Monastic Debate?

Monastic debate is a contemplative debating practice in which there is a large variability in the level of focus, as well as the reported emotions. Debating always involves at least one challenger and one defender. The challenger is standing and free to move while the defender sits for the entire duration of the debate. The debate has a strong formal structure, which is shown in Fig. 1. The goal for the challenger is to make the defender contradict one of the things he has agreed to before. The monastic debate motivate the debaters to improve reasoning abilities and memorization. It may also help them improve emotion regulation as debates can sometimes include teasing and insults to draw the defender out of their concentration and composure. More details about the debate can be found in [2]. In general, this shows that monastic debate is a fertile ground for EEG studies in ecologically valid contexts.

Objectives

We have the following two objectives for this study:

1. Determining whether EEG data can be collected in a real life situation and still render good detection of mental states using machine learning algorithms?
2. Is the classifier trained using machine learning on one set of data general enough to predict the cognitive states in another set of the data, when those data been acquired in different time frames and recording systems?

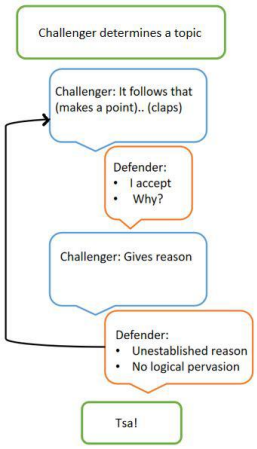


Figure 1: Structure of the debate

Dataset Description

The data consists of the EEG signals recorded from both the debaters simultaneously and the video recordings of the debate. Data has been collected in two sessions, one in 2017 (55 debates) and another in 2019 (46 debates) by two teams and two EEG recording systems (BrainVision actiCAP and Biosemi, respectively). Each debate has been labeled by at least three senior monks after watching the videos using the BORIS ethological observation software [3]. Ratings of focus and distraction were combined using the 'majority wins' rule.

Pre-Processing And Methodology

The EEG data were downsampled from 512 Hz to 256 Hz followed by the application of a band pass filter of 0.5 Hz - 40 Hz to remove low- and high-frequency artifacts. Next, Independent Component Analysis was done to clean the data of muscle and eye artifacts. Daubechies 4 wavelet transform was used to extract brain waves, namely delta (0.5-4 Hz), theta (4-9 Hz), alpha(9-14 Hz), beta (14-28 Hz), and gamma (28-40 Hz) from the pre-processed data. T-tests were used to identify channels and frequency bands that significantly distinguished between focus and distraction episodes identified by the raters. Subsequently, random forest classifiers were used to determine whether focus and distraction states could be detected on a single-trial level, and whether these would generalize across datasets.

Results

Statistical differences between focus and distraction

To check if average differences between the two cognitive states exist, the average raw EEG signals of all the participants for focused and distracted states were plotted along with the difference in the means. Fig. 2 and Fig. 3 show that channels 'FP_1', 'F_8', 'FP_2', 'Cz', 'O_2', 'TP_9', 'T_7', and 'P_7' show significant? differences.

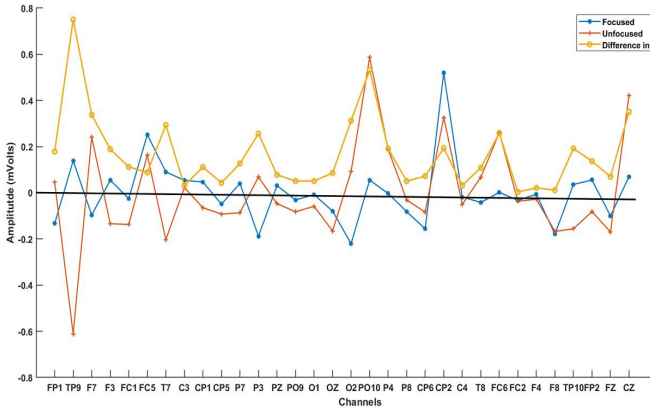


Figure 2: Raw EEG signals as a function of channel for 2017 dataset with focused (blue), distracted (orange) and their difference (yellow).

In addition to the raw EEG data, t-tests showed significant differences between focus and distraction in the delta band for the 2017 dataset, and in delta, theta alpha, and beta, bands for the 2019 dataset.

Machine Learning

To examine whether these differences could be observed on a single-trial level, we used a random forest classifier. Our random forest classifier consisted of 20 decision trees with maximum depth of each tree of 40 for the 2017 dataset. The accuracy obtained is shown in Fig. 4. The accuracy obtained is shown in Fig. 4. An accuracy of 79%, 97%, 93%, 99%, 88% and 99% was obtained for 2019 dataset in the raw, alpha, beta, delta, gamma and theta waves respectively using a single decision tree.

Conclusion and Future Work

Our results show that focus and distraction can be distinguished in EEG data collected in real life scenarios using statistical analysis and machine learning. In the future we will focus on determining if more subtle states like emotions can be detected and classified using this dataset as well.

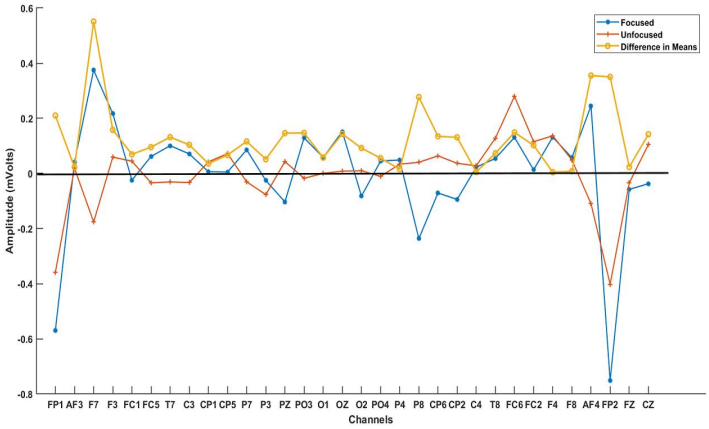


Figure 3: Raw EEG signals as a function of channel for the 2019 dataset with focused (blue), distracted (orange) and their difference (yellow).

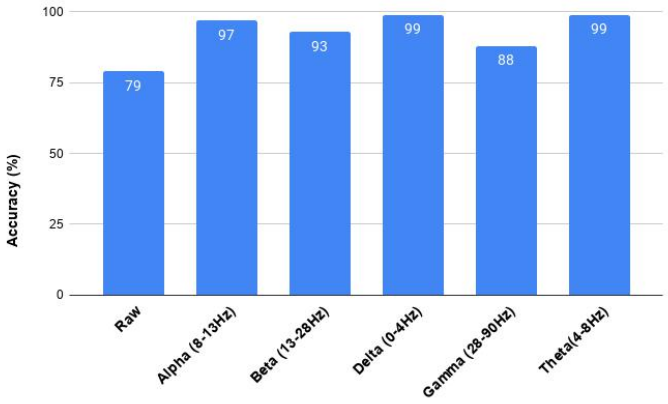


Figure 4: Accuracy obtained for 2017 dataset

References

Jin, C. Y., Borst, J. P., & van Vugt, M. K. (2019). Predicting task-general mind-wandering with EEG. *Cognitive, Affective, & Behavioral Neuroscience*, *19*(4), 1059-1073.

van Vugt, M. K., Pollock, J., Johnson, B., Gyatso, K., Norbu, N., Lodroe, T., & Lobsang, J. (2020). Inter-brain synchronization in the practice of Tibetan monastic debate. *Mindfulness*, 1-15.

http://www.boris.unito.it/pages/download.html

Modeling cross-language structural priming in sentence production

Yung Han Khoe (yhkhoe@protonmail.com)
CLS, Radboud University
Nijmegen, The Netherlands

Chara Tsoukala (x.tsoukala@gmail.com)
CLS, Radboud University
Nijmegen, The Netherlands

Gerrit Jan Kootstra (g.kootstra@let.ru.nl)
CLS, Radboud University
Nijmegen, The Netherlands

Stefan Frank (s.frank@let.ru.nl)
CLS, Radboud University
Nijmegen, The Netherlands

Abstract

A central question in the psycholinguistic study of multilingualism is how syntax is shared across languages. We implement a model to investigate whether error-based implicit learning can provide an account of cross-language structural priming. The model is based on the Dual-path model of sentence-production (Chang, 2002). We implement our model using the Bilingual version of Dual-path (Tsoukala, Frank, & Broersma, 2017). We answer two main questions: (1) Can structural priming of active and passive constructions occur between English and Spanish in a bilingual version of the Dual-path model? (2) Does cross-language priming differ quantitatively from within-language priming in this model? Our results show that cross-language priming does occur in the model. This finding adds to the viability of implicit learning as an account of structural priming in general and cross-language structural priming specifically. Furthermore, we find that the within-language priming effect is somewhat stronger than the cross-language effect. In the context of mixed results from behavioral studies, we interpret the latter finding as an indication that the difference between cross-language and within-language priming is small and difficult to detect statistically.

Keywords: cross-language structural priming; multilingualism; sentence production; syntax; dual-path model

Introduction

Psycholinguistic studies investigating syntax in both monolingual as well as multilingual speakers are often based on the structural priming paradigm. Structural priming is the tendency of speakers to reuse syntactic structures that they have previously encountered. In the study by J. K. Bock (1986) that introduced the paradigm, participants were more likely to use a passive target sentence (e.g., "The church is being struck by lightning") after repeating a passive sentence ("The referee was punched by one of the fans") than after repeating an active prime sentence ("One of the fans punched the referee"). Over 15 years ago, a number of studies showed that structural priming also occurs between two different languages (Hartsuiker, Pickering, & Veltkamp, 2004; Loebell & Bock, 2003; Meijer & Fox Tree, 2003).

Several studies have found no difference between the strength of within-language and cross-language structural priming (Hartsuiker, Beerts, Loncke, Desmet, & Bernolet, 2016; Kantola & van Gompel, 2011; Schoonbaert, Hartsuiker, & Pickering, 2007). In contrast, Cai, Pickering, Yan, and Branigan (2011) and Bernolet, Hartsuiker, and Pickering (2013) did find a stronger within-language than cross-language structural priming effect. This quantitative difference was accounted for by Bernolet et al. (2013) under the assumption that less proficient speakers of the second language (L2) had not yet developed syntactic representations that were shared across languages, or at least not for the syntactic structure under investigation. This would suggest that a prerequisite for equally strong within- and cross-language structural priming is that speakers are highly proficient in both languages.

Competing theoretical accounts of structural priming have been proposed. In the theoretical model introduced by Pickering and Branigan (1998), the residual activation of syntactic representations and combinatorial nodes leads to repeated use of particular syntactic representations. A bilingual version of this residual activation account was proposed by Hartsuiker et al. (2004). An alternative account explains structural priming as a form of error-based implicit learning (Chang, Dell, & Bock, 2006). According to this account, error-based learning causes changes in the extent to which different syntactic structures are expected to occur. When a prime sentence is processed, the connections associated with its syntactic structure are strengthened, making that structure's occurrence more expected. This learning mechanism affects the production of the target sentence as it increases the likelihood of producing the same structure. In this account, structural priming is therefore regarded as a long-lasting effect. Support for this view comes from a large number of studies that have demonstrated that structural priming can last over time and persists over the processing of other sentences (K. Bock & Griffin, 2000; Boyland & Anderson, 1998; Branigan, Pickering, Stewart, & McLean, 2000; Hartsuiker & Kolk, 1998; Huttenlocher, Vasilyeva, & Shimpi, 2004; Saffran & Martin, 1997).

Here, we investigate whether implicit learning can also account for cross-language priming. We do so by taking a well-known monolingual model of structural priming, and extending it to the bilingual case.

The Dual-path model

Different implemented cognitive models of monolingual structural priming have been introduced by Chang (2002), Malhotra (2009), and Reitter, Keller, and Moore (2011). We use Chang's (2002) Dual-path model, that explains a wide range of sentence production phenomena in a number of different languages.

Dual-path is an implicit learning model of sentence production. It is a connectionist model which is based on the Simple Recurrent Network (SRN; Elman, 1990) architecture. The first pathway in the model is the sequencing system, that learns how words are ordered in a sentence, while the second pathway acquires meaning-to-word-form mappings. Dual-path has been used to investigate monolingual structural priming in English (Chang et al., 2006) and German (Chang, Baumann, Pappert, & Fitz, 2015). Both of these studies demonstrated structural priming in the model, and thus provide support for the implicit learning account.

The model has also demonstrated the potential to account for experimental data from various second language acquisition and production studies. A Korean-English bilingual Dual-path model was used to examine the interaction between the effect of the age of acquisition and input factors, such as length of exposure, on second-language sentence production (Janciauskas & Chang, 2018). A Spanish-English bilingual version of this model was recently developed to investigate cross-linguistic transfer (Tsoukala et al., 2017) and code-switching (Tsoukala, Frank, van den Bosch, Valdéz Kroff, & Broersma, 2019). So far, no studies have been reported that demonstrate cross-language structural priming in the model.

The present study

We perform a computational modeling experiment to further test the viability of implicit learning as an account of structural priming in general and cross-language structural priming specifically. We do this by ascertaining whether cross-language structural priming can occur in the Dual-path model. We simulate cross- and within-language priming of actives and passives, using artificial versions of Spanish and English. Furthermore, we investigate if cross-language priming differs quantitatively from within-language priming in the model.

We expect cross-language structural priming to occur, because cross-language structural priming has been experimentally demonstrated by Hartsuiker et al. (2004) in adults, and by Vasilyeva et al. (2010) in children, for the languages and syntactic structures used in the present work. Additionally, as mentioned above, within-language priming has been shown to occur in the model (Chang et al., 2006, 2015). Finally, a bilingual version of the Dual-path model has demonstrated the ability to code-switch, without being exposed to code-switched language (Tsoukala et al., 2019) and code-switching has been interpreted as an indication that syntax is shared between languages (Kootstra, Van Hell, & Dijkstra, 2010; Loebell & Bock, 2003).

Assuming the model does display cross-language structural priming, we have no strong expectation of whether or not it will differ in strength from within-language priming. However, we aim to meet the suggested prerequisite for equivalent within- and cross-language priming effects (Bernolet et al., 2013) by simulating balanced bilingual speakers, who are equally proficient in both languages.

Method

Model

To simulate participants in a cross-language priming experiment, we trained the Bilingual Dual-path model[1] (Figure 1) to simulate simultaneous Spanish-English bilinguals, who start acquiring both Spanish and English from infancy.

The Bilingual Dual-path model is a modified version of the original Dual-path model (Chang, 2002). The training input to the model consists of sentences in an artificial language that are paired with messages that encode their meaning (see examples below, under: Artifical languages). The model learns to convert a message into a sentence by predicting the sentence word by word. A difference between the Dual-path architecture and other Recurrent Neural Networks is that the network has connections with fixed weights between concepts and roles of the message to be expressed.

Artificial languages Both artificial languages[2] we used include the same twelve sentence types: Animate intransitive, Animate with-intransitive, Inanimate intransitive, Locative, Transitive (in active or passive form), Cause-motion, Transfer dative (in prepositional object (PO) form), Benefactive dative (in PO form), Benefactive transitive, State-change, and Locative alternation[3]. The two languages together have 275 unique lexical items. In addition to nouns, verbs, adjectives, determiners, and prepositions, these lexical items include inflectional morphemes such as a past tense marker (Spanish: '-pas'; English: '-pst') and a past participle marker (Spanish: '-prf'; English: '-par'). The message semantics contain 121 concepts, and 7 thematic roles. These numbers differ somewhat from those that were preregistered (see Section Pre-registered analysis). This is because we made small adjustments to the auxiliary verbs and inflectional morphemes of the artificial languages. None of these changes lead to different answers to our research questions. Only singular verbs, pronouns, nouns, and adjectives were used. Verbs and pro-

[1]The Bilingual Dual-path model can be downloaded from: https://github.com/xtsoukala/dual_path

[2]The files that the model requires to generate the artificial language input, and the input for the priming experiment can be found here: https://osf.io/pm6f9/

[3]Examples for these sentence types can be found in Chang et al. (2006)

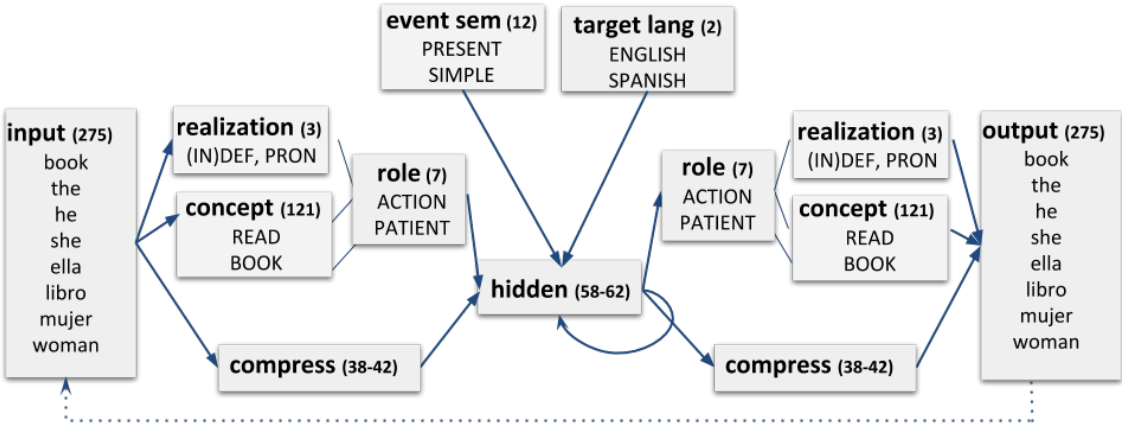


Figure 1: **Bilingual Dual-path, the model used in our priming experiment.** The model is a next-word prediction model that converts messages into sentences. It is an SRN-based model (the lower path, via the 'compress' layers) that is augmented with a semantic stream (upper path) that contains information about concepts, thematic roles, event semantics, and the target language. The number of units per layer are shown in parentheses. The numbers of units for the hidden and compress layers vary across simulations. (Figure adapted from Tsoukala et al. (2017))

nouns were always in third person form.

Because our aim is to verify the possibility of structural priming between different languages, we designed artificial versions of English and Spanish that maximize the likelihood of revealing an effect. If a structure is produced very frequently irrespective of priming, a small increase caused by priming might not result in a detectable effect. We addressed this issue by using balanced frequencies of the structures under investigation. This means that actives (see examples 1, 2 below) and passives (examples 3, 4) occur with the same frequency in the training input we provide the model. For similar reasons, we model balanced bilingual speakers by training the model on both languages from the beginning and on almost equal numbers of sentences in the two languages, that randomly deviate only marginally.

In the training and test input, each message that can be expressed using two different syntactic structures has a strong bias towards one of those structures. This was done by creating differences in activation based on how each structure emphasizes thematic roles in the sentence. Biasing towards an active sentence (1, 2), for example, was done by giving the agent a higher activation (X:1) than the patient (Y:0.5 or Y:0.75). In the same way, a bias towards a passive sentence (3, 4) was achieved with a higher activation for the patient (Y:1), than for the agent (X:0.5 or X:0.75). Similarly to Chang et al. (2006), we gave the target messages in the priming experiment a weaker bias than we used in the training and test input by giving the de-emphasized roles an activation of 0.95 instead of 0.5 or 0.75.

1. Spanish Active: el padre romper -pas la botella .
   X = def, FATHER, M;
   ACTION-LINKING = BREAK;
   Y = def, BOTTLE;
   EVENT-SEM = **X:1, Y:0.5**, PAST,
   SIMPLE, ACTION-LINKING;
   TARGET-LANG = es

2. English Active: the father break -pst the bottle .
   [...];
   EVENT-SEM = **X:1, Y:0.5**, PAST,
   SIMPLE, ACTION-LINKING;
   TARGET-LANG = en

3. Spanish Passive: la botella es romper -prf por el padre .
   [...];
   EVENT-SEM = **X:0.5, Y:1**, PAST,
   SIMPLE, ACTION-LINKING;
   TARGET-LANG = es

4. English Passive: the bottle is break -par by the father .
   [...];
   EVENT-SEM = **X:0.5, Y:1**, PAST,
   SIMPLE, ACTION-LINKING;
   TARGET-LANG = en

Training and testing model accuracy A set of 8,000 unique message-sentence pairs was generated for each model participant. 80% of these sentences were used for training, while 20% were set aside for testing the accuracy of the model. Following Chang et al. (2006), the message was excluded from 25% of training pairs. The models iterated over their training sets 20 times. After each of these 20 epochs, model accuracy was tested using the test set. The training set was shuffled at the beginning of each epoch.

Model configuration The models have a number of hidden layer units that was sampled from a uniform distribution between 58 and 62, and a number of compress layer units sampled from a uniform distribution between 38 and 42. The fixed weight value for concept–role connections was sampled from a uniform distribution between 13 and 17. The sentences are approximately equally divided over the two languages, where the language percentage of English was sampled from a uniform distribution between 48 to 52% and the rest was Spanish. Other than this, we used the model's default settings.

Priming experiment

Model participants Some of the model participants we trained did not successfully learn the artificial languages. We therefore trained 120 models and selected the 80 model participants with the highest meaning accuracy (i.e. percentage of grammatically correct sentences that convey the target message without any additions, over all test sentences). The accuracy scores for these models varied from 70.5% to 87.9%. A supplementary analysis includes all 120 model participants.

Experimental trials Independent of the training and test sets, a single set of experimental trials was generated that was used to perform the priming experiment on all of the model participants. Each trial consisted of a combination of a unique prime sentence and a unique target message that did not have any semantic overlap in terms of their verb, agent, and patient. With two types of LANGUAGE COMBINATION trials and two types of PRIME LANGUAGE trials, we had four possible combinations of prime- and target-language: English-English, Spanish-Spanish, Spanish-English, and English-Spanish. We had equal numbers of these four language combinations, which in turn means that there were equal numbers of within- and cross-language trials. We also had equal numbers of trials with active and passive primes, and equal numbers of trials with active- and passive-bias target-messages. The two types of trials for PRIME STRUCTURE, LANGUAGE COMBINATION, PRIME LANGUAGE, and TARGET-MESSAGE BIAS combine for a total of 16 different trial types. We had 50 prime-target combinations that all occurred as each of the 16 different trial types. This means that each experiment consisted of 800 trials.

Procedure The priming experiment was performed on the models after 20 training epochs. As was done in Chang et al. (2006) and Chang et al. (2015), we presented the models with prime sentences without a message, and with learning turned on in the model. After each prime, a response was elicited from the model by presenting it with a target message.

After each trial, the connection weights were reset to the values they had before starting the priming experiment. The state in which the model encounters each trial was thus the same for all of the trials, hence, there was no between-trial priming or any other learning effects during the experiment. This means that the order of the trials did not need to be (pseudo-)randomized across model participants.

For the priming experiment, the learning rate was set to 0.2. In our pre-registration we reported that a learning rate of 1.2 would be used there, but this was an error in the pre-registration. The exploratory experiment on within-language priming in fact used a learning rate of 0.2 during the priming experiment, and this was also the intended learning rate for the pre-registered experiment. This difference with the preregistration does not increase the probability of finding an effect. If anything, a higher learning rate would have resulted in a larger priming effect (Chang et al., 2006).

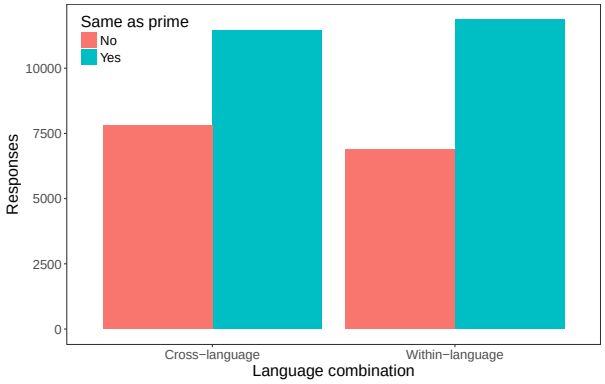


Figure 2: **Total number of responses that had either the same structure as the prime or the alternative structure, split by within- or cross-language trials.** The plot shows that there were more same structure than alternative structure responses for both LANGUAGE COMBINATION conditions. However, the difference was larger for within-language trials.

Results

Our analysis only included those responses that correctly conveyed the target message, either with an active or a passive structure. However, we disregarded errors involving definiteness of articles or missing periods. This means we included 60% of the responses on cross-language trials, and 59% of responses on within-language trials. On cross-language trials, model participants produced 11,451 sentences (59% of sentences) with the same structure as the prime, while producing 7,802 sentences (41%) with the alternative structure. On within-language trials, 11,856 responses (63%) had the same structure as the prime, whereas 6,877 responses (37%) had the alternative structure. Figure 2 shows that there were more same-structure than alternative-structure responses for both LANGUAGE COMBINATION conditions, but that the difference was larger for within-language trials.

Pre-registered analysis

As pre-registered[4], we analyzed the data from our experiment with a Bayesian logistic mixed-effects model, using a logit link function, using the function brm from the package brms (Bürkner et al., 2017; Bürkner, 2018, version 2.12.0) in R (R Core Team, 2013, version 3.5.1). The model predicts a binary dependent variable, SAME AS PRIME, that indicates whether the sentence structure that the model produced and the structure of the prime sentence were the same (1), or different (0). In addition to the predictor of interest, LANGUAGE COMBINATION (Cross-language = 0, Within-language = 1), the model includes three contrast-coded covariates: PRIME STRUCTURE (Active = −0.5, Passive = 0.5), TARGET-MESSAGE BIAS (Active = −0.5, Passive = 0.5), and PRIME LANGUAGE (English = −0.5, Spanish = 0.5). We fit random intercepts for items and model participants, as well as by-item and by-participant random slopes for LANGUAGE COMBINATION. Regularizing priors were used in all our models, which give a minimal amount of information with

[4]The pre-registration can be accessed here: `https://aspredicted.org/728mn.pdf`

Table 1: Summary of the fixed effects in the Bayesian logistic mixed-effects model ($N = 37,946$). For each predictor are shown its estimate with 95% Bayesian credible interval and the posterior probability that the estimate is positive.

Predictor	Estimate	95% CrI	P(Estimate > 0)
INTERCEPT	0.49	[0.38, 0.60]	1.00
LANGUAGE COMBINATION	0.23	[0.12, 0.34]	1.00
PRIME LANGUAGE	0.16	[0.01, 0.29]	0.99
PRIME STRUCTURE	0.84	[0.74, 0.94]	1.00
TARGET-MESSAGE BIAS	-0.31	$[-0.41, -0.22]$	0.00
LANGUAGE COMBINATION $\times$ PRIME LANGUAGE	-0.06	$[-0.25, 0.13]$	0.27
PRIME STRUCTURE $\times$ TARGET-MESSAGE BIAS	2.98	[2.79, 3.19]	1.00

the objective of yielding stable inferences. Prior means were 0, and did thus not bias towards specific effects. The standard deviations for the priors that we used for the predictors are based on the effect sizes that resulted from our preliminary analysis of within-language priming.

Unfortunately, even when using a large number of 16,000 iterations, 12 chains, and a high value of 0.99 for the adapt_delta parameter, this model did not result in valid and reliable parameter estimates. This was apparent from the large number of divergent transitions after warmup, and the low Bulk and Tail Effective Sample Sizes (ESS) (`https://mc-stan.org/misc/warnings.html`). Analysis of the model output revealed that the ESS values were specifically related to the estimates of the by-item random slopes for LANGUAGE COMBINATION. In addition, the credible interval (CrI) for these estimates were consistently close to zero across different numbers of iterations and chains, and different values for the adapt_delta parameter.

Adjusted analysis

Because of the reliability issues in the pre-registered regression model we removed the by-item random slopes and their correlations from the model, while leaving it unchanged otherwise. The resulting model yielded valid and reliable results, as evidenced by the absence of the type of warnings that the pre-registered model resulted in. Note, however, that both models revealed the same pattern of results that lead to the same answers to our research questions.

The regression analysis results are summarized in Table 1. The positive intercept, with a credible interval far from zero, shows a clear priming effect at the reference level (i.e., cross-language) of the LANGUAGE COMBINATION predictor. We interpret this as strong evidence for cross-language priming in the Dual-path model. The positive estimate for the LANGUAGE COMBINATION predictor, with a credible interval that does not cross zero, indicates that the within-language priming effect is stronger in the Dual-path model than the cross-language priming effect.

Discussion

The results of our experiment reveal a clear and strong cross-language structural priming effect. We thus provide evidence for the viability of implicit learning as an account of cross-language structural priming. In turn, our finding provides support for the implicit learning model implemented in Dual-path, as an account of structural priming in general. We should note, however, that this finding does not provide evidence against other implemented models of structural priming. The hybrid model introduced by Reitter et al. (2011), for example, also predicts cross-language structural priming. Fortunately, a way to empirically distinguish between this account and the Dual-path account is available. The former account predicts that priming will not occur between structures in different languages that do not have the same word order. The Dual-path account, on the other hand, does not seem to rule out such a priming effect.

We also find slightly stronger within- than cross-language structural priming. As a number of behavioral studies failed to find a significant difference between cross- and within-language priming, this could suggest that our model provides an insufficiently adequate account of structural priming in this respect. However, other studies did find differences in the strength of cross-language and within-language priming, and these differences have been explained as resulting from participants' proficiency differences between the two languages. It might therefore be the case that we did not succeed in simulating sufficiently balanced bilinguals, although this is unlikely given the way our models were trained.

However, a simple account that could explain both the available behavioral results and our modeling results seems plausible. The presence of a small difference between cross- and within-language priming, that is hard to detect statistically in highly proficient bilinguals, is consistent with the human data. This difference could become clearer if either proficiency differences increase, or if experimental methods are applied that are more likely to detect small effects. In addition, the absence of a significant effect is not proof that an effect does not exist (Vasishth & Nicenboim, 2016).

In two of the four experiments reported on by Hartsuiker et al. (2016), for example, within-language priming was stronger than cross-language, even though this difference was not found to be significant. In the other two experiments, cross-language priming was either stronger than within-language priming or it depended on the prime lan-

guage, but in any case none of the differences were statistically significant. In the first experiment conducted by Kantola and van Gompel (2011), within-language priming was non-significantly stronger than cross-language priming, whereas in the second experiment within-language priming was non-significantly weaker than cross-language priming. However, the difference in strength was larger in the first than in the second experiment. The results reported by Schoonbaert et al. (2007) and re-analysed by Hartsuiker et al. (2016); Hartsuiker and Bernolet (2017) showed non-significantly stronger within-language than cross-language priming when there was no semantic overlap between verbs in the prime and target sentences, as was the case in our comparison. When prime and target verbs were identical or translation-equivalent, however, Schoonbaert et al. (2007) found that within-language priming was significantly stronger than between-language priming. Overall, these results do not provide strong evidence that cross-language and within-language priming are equally strong.

Our analysis does reveal a clear effect of language combination. However, our experiment is relatively large in terms of the number of model participants and especially large in terms of the number of trials per participant. We performed a post-hoc analysis on a subset of our data that is more comparable to (but still larger than) the size of behavioral experiments. From the original 80 model participants, we excluded the 20 highest performing and the 20 lowest performing participants. We reduced the number of trials from 800 to 208 per participant while keeping the same distribution across conditions. The positive intercept resulting from this analysis still reveals a clear cross-language priming effect: Estimate = 0.38, 95% CrI = [0.19, 0.57]. In contrast, the LANGUAGE COMBINATION predictor now has a credible interval that crosses zero (Estimate = 0.19, 95% CrI = [-0.03, 0.43]). It therefore does not provide strong evidence anymore for a difference between cross- and within-language priming. If the model gives an approximately correct estimate of the difference between within- and cross-language priming, we cannot expect human studies to reveal that difference with the amount of data they have available. This is especially true if we consider that human data generally has a lower signal-to-noise ratio than modeling data.

It could be costly to conduct an experiment with a large number of participants to verify that within-language priming is stronger than cross-language priming in balanced bilinguals. Likewise, increasing the number of experimental trials might cause concentration problems in participants. A way to address this might be to conduct the study as a large online experiment.

Further work

Relative proficiency in the two languages involved in cross-language structural priming can influence the strength of the priming effect. Now that we have established that cross-language priming occurs in the Dual-path model, a fruitful direction for future research will be to explore the relationship between second language proficiency and structural priming between languages.

As argued by, for example, Bernolet, Hartsuiker, and Pickering (2009), structural priming could be a phenomenon that takes place at different levels (e.g., information structure and syntactic structure), and syntactic alternations are different in the extent to which these levels play a role. To reach a deeper understanding of structural priming, it is therefore important to extend our modeling to further syntactic alternations, such as datives and genitives.

References

Bernolet, S., Hartsuiker, R. J., & Pickering, M. J. (2009). Persistence of emphasis in language production: A cross-linguistic approach. *Cognition*, *112*(2), 300–317.

Bernolet, S., Hartsuiker, R. J., & Pickering, M. J. (2013). From language-specific to shared syntactic representations: The influence of second language proficiency on syntactic sharing in bilinguals. *Cognition*, *127*(3), 287–306.

Bock, J. K. (1986). Syntactic persistence in language production. *Cognitive Psychology*, *18*(3), 355–387.

Bock, K., & Griffin, Z. M. (2000). The persistence of structural priming: Transient activation or implicit learning? *Journal of Experimental Psychology: General*, *129*(2), 177.

Boyland, J. T., & Anderson, J. R. (1998). Evidence that syntactic priming is long-lasting. In *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society* (Vol. 1205).

Branigan, H. P., Pickering, M. J., Stewart, A. J., & McLean, J. F. (2000). Syntactic priming in spoken production: Linguistic and temporal interference. *Memory & Cognition*, *28*(8), 1297–1302.

Bürkner, P.-C. (2018). Advanced Bayesian Multilevel Modeling with the R Package brms. *The R Journal*, *10*(1), 395–411.

Bürkner, P.-C., et al. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, *80*(1), 1–28.

Cai, Z. G., Pickering, M. J., Yan, H., & Branigan, H. P. (2011). Lexical and syntactic representations in closely related languages: Evidence from Cantonese–Mandarin bilinguals. *Journal of Memory and Language*, *65*(4), 431–445.

Chang, F. (2002). Symbolically speaking: A connectionist model of sentence production. *Cognitive Science*, *26*(5), 609–651.

Chang, F., Baumann, M., Pappert, S., & Fitz, H. (2015). Do lemmas speak German? a verb position effect in German structural priming. *Cognitive Science*, *39*(5), 1113–1130.

Chang, F., Dell, G. S., & Bock, K. (2006). Becoming syntactic. *Psychological Review*, *113*(2), 234.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Hartsuiker, R. J., Beerts, S., Loncke, M., Desmet, T., & Bernolet, S. (2016). Cross-linguistic structural priming in multilinguals: Further evidence for shared syntax. *Journal of Memory and Language*, *90*, 14–30.

Hartsuiker, R. J., & Bernolet, S. (2017). The development of shared syntax in second language learning. *Bilingualism*, *20*(2), 219.

Hartsuiker, R. J., & Kolk, H. H. (1998). Syntactic persistence in dutch. *Language and Speech*, *41*(2), 143–184.

Hartsuiker, R. J., Pickering, M. J., & Veltkamp, E. (2004). Is syntax separate or shared between languages? cross-linguistic syntactic priming in Spanish-English bilinguals. *Psychological Science*, *15*(6), 409–414.

Huttenlocher, J., Vasilyeva, M., & Shimpi, P. (2004). Syntactic priming in young children. *Journal of Memory and Language*, *50*(2), 182–195.

Janciauskas, M., & Chang, F. (2018). Input and age-dependent variation in second language learning: A connectionist account. *Cognitive Science*, *42*, 519–554.

Kantola, L., & van Gompel, R. P. (2011). Between-and within-language priming is the same: Evidence for shared bilingual syntactic representations. *Memory & Cognition*, *39*(2), 276–290.

Kootstra, G. J., Van Hell, J. G., & Dijkstra, T. (2010). Syntactic alignment and shared word order in code-switched sentence production: Evidence from bilingual monologue and dialogue. *Journal of Memory and Language*, *63*(2), 210–231.

Loebell, H., & Bock, K. (2003). Structural priming across languages. *Linguistics*, *41*(5; ISSU 387), 791–824.

Malhotra, G. (2009). *Dynamics of structural priming*. Unpublished doctoral dissertation, University of Edinburgh.

Meijer, P. J., & Fox Tree, J. E. (2003). Building syntactic structures in speaking: A bilingual exploration. *Experimental Psychology*, *50*(3), 184.

Pickering, M. J., & Branigan, H. P. (1998). The representation of verbs: Evidence from syntactic priming in language production. *Journal of Memory and Language*, *39*(4), 633–651.

R Core Team. (2013). R: A language and environment for statistical computing.

Reitter, D., Keller, F., & Moore, J. D. (2011). A computational cognitive model of syntactic priming. *Cognitive Science*, *35*(4), 587–637.

Saffran, E. M., & Martin, N. (1997). Effects of structural priming on sentence production in aphasics. *Language and Cognitive Processes*, *12*(5-6), 877–882.

Schoonbaert, S., Hartsuiker, R. J., & Pickering, M. J. (2007). The representation of lexical and syntactic information in bilinguals: Evidence from syntactic priming. *Journal of Memory and Language*, *56*(2), 153–171.

Tsoukala, C., Frank, S., van den Bosch, A., Valdéz Kroff, J., & Broersma, M. (2019). Simulating Spanish-English code-switching: El modelo está generating code-switches. In *CMCL 2019: Workshop on Cognitive Modeling and Computational Linguistics.*

Tsoukala, C., Frank, S. L., & Broersma, M. (2017). "he's pregnant": Simulating the confusing case of gender pronoun errors in L2 English. In *the 39th Annual Meeting of the Cognitive Science Society (CogSci 2017)* (pp. 3392–3397).

Vasilyeva, M., Waterfall, H., Gámez, P. B., Gómez, L. E., Bowers, E., & Shimpi, P. (2010). Cross-linguistic syntactic priming in bilingual children. *Journal of Child Language*, *37*(5), 1047–1064.

Vasishth, S., & Nicenboim, B. (2016). Statistical methods for linguistic research: Foundational ideas–Part I. *Language and Linguistics Compass*, *10*(8), 349–369.

A Study in Activation: Towards a Common Lexicon and Functional Taxonomy in Cognitive Architectures

Sean Kugele (seankugele@gmail.com) and Stan Franklin (franklin.stan@gmail.com)
Department of Computer Science and Institute for Intelligent Systems, University of Memphis
Memphis, TN 38152 USA

Abstract

Activation has become a pervasive concept in many scientific disciplines, including cognitive and neural modeling, and AI. Unfortunately, its applications and functions are so broad and varied that it is difficult for practitioners to discuss the topic in precise and meaningful ways. This is particularly apparent in cognitive architectures, where a wider breadth of activation's utilities and forms have been explored. To help combat these terminological difficulties, and hopefully facilitate productive discourse and the development of future applications, we introduce (1) a *lexicon* of activation-related concepts, and (2) a *functional taxonomy* that enumerates many activation-related "design patterns" that have appeared in cognitive architectures. We demonstrate our taxonomy by applying it to the LIDA cognitive architecture, which includes one of the most varied and comprehensive adoptions of activation-related functionality.

Keywords: activation, cognitive architectures, LIDA

Introduction

The concept of "activation" has become commonplace in many scientific disciplines. In the neural sciences, it refers to "patterns of neural firing," measured either individually or collectively. In chemistry, it refers to the transition of a molecule to a state with an "increased probability" of a chemical reaction. And, in psychology, it has been used to refer to the "level of arousal or excitation" observed in an individual "as a whole" (Duffy, 1957).

Within the fields of artificial intelligence (AI) and cognitive modeling, activation describes an assortment of quantities and parameters, which have been used to implement a diverse range of functionality. Certainly, activation and "activation functions" have figured prominently in the development of artificial neural networks (ANNs). However, the full breadth of activation's utility and forms has come to fruition in the many cognitive architectures that have embraced, and expanded on, the concept. Unfortunately, these applications and functions are so broad and varied that it is difficult for practitioners to discuss the topic in precise and meaningful ways.

To get a better sense for this diversity, we reviewed seventy-eight cognitive architectures[1] in search of distinct activation-related concepts and themes. Thirty-three of these were found to use some form of activation. While a comprehensive survey of each cognitive architecture's use of activation is beyond the scope of this paper, we believe that we have succeeded in identifying the major concepts and themes. We have distilled these into a *lexicon* of activation-related terminology and a *functional taxonomy* that categorizes each theme with respect to its functional intent (that is, what activation affects, facilitates, or enables within a cognitive architecture). We believe that this is an essential first step towards standardizing notions of activation across cognitive architectures.

Among the cognitive architectures, LIDA (Franklin, et al., 2016) implements one of the most varied and comprehensive adoptions of activation. It uses activation to support nearly every module and process, and one or more activation parameters are associated with most (if not all) of its mental representations. Given this abundance of function and form, LIDA provides a plentiful source of examples, which we use to test our taxonomy's utility, and illustrate its taxonomic themes.

What is Activation?

Arguably, the earliest application of activation-related concepts in AI and cognitive modeling occurred in the context of connectionist models, such as artificial neural networks (ANNs). ANNs are biologically-inspired computational systems composed of "artificial neurons" (also called neural units) that are typically connected in layered architectures. Each neural unit performs a calculation (for example, a weighted sum) over its inputs, and the result is referred to as that neuron's "activation." Activations are then passed through "activation functions" (for example, unit step, sigmoid, or rectified linear) to determine a neuron's output (or response)[2]. These response values can "propagate" to connected neural units, where they are used as inputs to further computations. (This is also referred to as "spreading activation.") ANNs learn "distributed representations" corresponding to the *patterns of activation* induced in the network by various stimuli.

Cognitive architectures have expanded on these connectionist concepts, inventing a host of new mechanisms with their own distinct dynamics. To make sense of this variability, we introduce a basic lexicon of activation-related concepts, and then we review and categorize noteworthy applications of activation within cognitive architectures. This is then used as the catalyst for our

[1] This included most of the cognitive architectures mentioned in (Kotseruba & Tsotsos, 2018), as well as several that were not mentioned.

[2] In the literature, the output of an artificial neural unit is sometimes referred as the unit's "activation"; however, we use the convention that activation refers to the "internal state" resulting from a calculation over its inputs and weights that may be passed as input to an activation function.

"functional taxonomy," which is presented later in the paper.

Concepts and Terminology

Activations are typically implemented as continuous, scalar quantities, or vectors of such quantities (such as, "patterns of activation"). They must have an *activation source* ("how activation is acquired?"), an *activation target* ("what gets activated?"), and a *function(s)* in the system. An implicit or explicit *decay strategy* must also be provided. Optionally, an *activation spreading* mechanism can be implemented that propagates activation between one or more activation targets, and an *activation threshold* can be specified that requires a target's activation be above, or below, a specific value before initiating its associated functionality.

Activation Sources. Anything can be used as a source of activation. The only requirement is that it activates its target *consistently* with respect to some intended purpose. For example, if an activation parameter is intended as a measure of its target's "relevance", "urgency", "salience", "reliability", etc., then the activation source must generate activation in proportion to the target's current compatibility with that measure.

In practice, an *activation source* is often a (mathematical) function specified in terms of other activations or conceptual quantities. An example of this is ACT-R's (Anderson, et al., 2004) formula for determining the activation of a "chunk" (that is, a declarative unit of knowledge):

$$A_i = B_i + \sum_j W_j S_{ji},$$

where A_i is the ith chunk's activation, B_i is its *base-level activation*, W_j is the "attentional weighting" of the jth element in the "current goal," and S_{ji} is the *associative activation* between chunk i and its jth supporting element. In this context, base-level activation and associative activation are activation sources that determine the activation of a chunk. Base-level activation (B_i) has its own activation source, based on the function

$$B_i = \ln\left(\sum_j t_j^{-d}\right),$$

where t_j is the time that has passed since the jth retrieval of chunk i, and d is a fixed parameter that determines the shape of the learning/forgetting curve.

Activation Targets. "Knowledge" representations (semantic, perceptual, procedural, etc.) are the most common *activation targets* in cognitive architectures. However, other data structures, processes, modules, and even entire cognitive systems have been used. The only requirement is that the target is an entity whose function (or identity) can be meaningfully modulated (or determined) by an activation's value.

Decay Strategies. *Explicit* decay strategies are often implemented using (mathematical) functions that are periodically invoked for the purpose of decreasing the value of an activation variable over time. For example, DUAL (Kokinov, 1994) invokes an exponential decay function each time step to decrease the activation of its working memory elements (save for the most active, which is referred to as "the focus"). By contrast, *implicit* strategies often implement decay directly within activation sources; for example, in ACT-R's base-level activation equation (shown earlier), the contribution of the jth chunk retrieval "decays away as a power function (producing the power law of forgetting)" (Anderson, et al., 2004).

Spreading Activation. Many cognitive architectures allow activation to "propagate" between *associated activation targets*. For example, Copycat's (Hofstadter & Mitchell, 1995) long-term memory module, the Slipnet, supports activation spreading between its nodes. These nodes (representing concepts) serve as activation targets, and its links (associations) serve as conduits that allow the passing of activation between them. The "conceptual distance" between nodes, which is based on the activation of its links' labels, determines the amount of activation spread.

A more sophisticated example occurs in the Agent Network Architecture (ANA) (Maes, 1991), which features predecessor, successor, and conflictor links that allow "activation energy" to spread between, and accumulate in, different competence modules (for example, action and belief modules). Predecessor and successor links are *excitatory* (increasing the activations of their associated activation targets) while conflictor links are *inhibitory* (decreasing the activations of their associated activation targets). The magnitude of this increase or decrease is proportional to a (source) competence module's activation.

As a final example, 4CAPS (Just & Varma, 2007) supports activation spreading using weighted *condition-action* production rules that function like weighted links. These productions spread activation iteratively (that is, over multiple "cycles") from task-activated cortical "centers" to their associated declarative elements.

Activation Thresholds. Activation is often functionally "inert" (that is, its associated functionality is not invoked) unless its value crosses above or below an *activation threshold*. Such thresholds control the retrieval of ACT-R's chunks (Anderson, et al., 2004), the execution of ANA's competence modules (Maes, 1991), and the spreading of activation and updating of network weights (associative learning) in LEABRA (O'Reilly, 1996).

Activation thresholds in cognitive architectures can be thought of as generalizations of the binary threshold functions (unit step functions) that appeared in early connectionist networks, such as perceptrons (Rosenblatt, 1958). However, while the function of a perceptron's activation threshold (θ) is limited to modulating its units' "all-or-nothing" output signals ($+1$ if a unit's activation $\geq$

θ; 0 otherwise), the range of functionality that can be modulated in cognitive architectures is seemingly limitless.

Activation's Functions in Cognitive Architectures

Having established a basic vocabulary of activation-related concepts, we now present the major activation-related functional themes that have appeared in cognitive architectures.

Access to Mental Representations. One of the most common uses of activation in cognitive architectures has been to influence global, module-specific, or process-specific access to mental representations. Activation, in this context, can be interpreted as specifying the current, context-specific *relevance* of mental representations, allowing cognitive resources to be focused on representations that matter most at a given moment. For example, in ACT-R, activation controls both the probability and timing of declarative memory (that is, "chunk") retrieval. In Soar (Laird, 2012), activation biases the retrieval of episodic memories. And, in CERA-CRANIUM (Arrabales, Ledezma, & Sanchis, 2009), the priority of percept processing is determined by activation, where those with the lowest activations are not processed at all.

Removal of Mental Representations. Activation has been used to modulate the purging and/or pruning[3] of short-term and long-term mental representations. For example, Soar removes working-memory elements when their activations have decayed below some fixed (removal) threshold (Laird, 2012). These representations are still available in long-term semantic memory for later retrieval, but the system has determined that they are no longer directly relevant to its current task (based on their activations). In other words, these representations are "gone, but not forgotten." We refer to this as "bounded" removal. Representations can also be removed "globally," such as occurs when the base-level activations associated with LIDA's (Franklin, et al., 2016) long-term memory representations (declarative, perceptual, procedural, etc.) decay below a removal threshold. In these cases, the representations are no longer available for use or retrieval. That is, they have been "forgotten."

This functional theme complements activation's use as an *access modulator*, and both uses often appear together in cognitive architectures. Jointly, they can be said to determine the "availability" of mental representations.

Informational Content. Cognitive architectures have used *patterns of activation* as "informational content." These activation patterns must somehow represent the sensory, perceptual, and/or conceptual essences of experiences, introspections, etc. This theme is exemplified in ART (Grossberg, 1999), where patterns of activations are stored as short-term or long-term memory "traces." Shanahan's (2006) brain-based implementation of Global Workspace Theory (Baars, 1988) also makes extensive use of patterns of activation as mental representations.

Associative Dynamics. Activation has been used to represent the time-varying, context-sensitive, strength of associations between mental representations. Here, activation can be interpreted as associative weights, or modulators of associative weights, whose values are influenced by situational context or prior experiences. ACT-R's "associative activations" are one example of this. Another example occurs in Copycat's Slipnet, where the conceptual distances between its nodes are based on the activations of its links' "labels." Concepts (that is, nodes) that are closer in conceptual distance (that is, have higher link label activations) are more likely to "slip" into one another, and be treated as analogous concepts.

System Dynamics. Activation can locally or globally modulate "how" cognitive operations are performed. In this context, activation could be viewed as representing dynamic dispositions, temperaments, or moods, and the notion of "activation as arousal" (see (Duffy, 1957)) is consistent with this theme. An example of this from a cognitive architecture is Copycat's "temperature," which is described by Hofstadter and Mitchell (1995) as a variable that "monitors the stage of processing, and helps to convert the system from its initial largely bottom-up, open-minded mode to a largely top-down, closed-minded one."

Measures of Intensity or Degree. Activation often represents a graded measure of some quantity with a clear semantic interpretation (for example, "reliability") that fluctuates in intensity over time. Due to the conceptual interpretability of these measures, they often serve as activation sources that modulate other cognitive functions. An example of activation as a *measure of intensity* appears in Leabra, where network activations represent "graded (continuous) states of truth-value" that estimate "the degree to which [a hypothesis] is believed to be true by the network" (O'Reilly, 1996). Another example occurs in LIDA, where the activations associated with LIDA's feeling nodes quantify an agent's current "liking" or "disliking" of a stimulus.

Process Scheduling. Activation has been used to determine or influence the execution of events, tasks, processes, and modules. The CopyCat architecture contains a module called the Coderack that serves as a pool of "codelets[4]." Each codelet is associated with an "urgency" value that is a function of the activation patterns in the Slipnet. These are used by the Coderack to determine the *probability* that a particular codelet will be selected for execution. Activation,

[3] Pruning refers to the extraction of a mental representation from a data structure (like a tree or associative network) that requires additional structural maintenance (such as the removal of links or "dangling" associations) to purge the targeted item.

[4] Codelets are processes that function as simple agents with the ability to find, create, or destroy structures in Copycat's Workspace.

in this context, can be viewed as *influencing system dynamics* through the immediate or future execution of some cognitive process. Another example occurs in DiPRA (Pezzulo, 2009), which contains an "energy pool" from which its modules receive activation at the beginning of each execution cycle. Since activation is required for module execution, if the energy pool is depleted in a given cycle, then one or more modules may have to wait until a later cycle to execute.

A Functional Taxonomy

In this section, we present our "functional taxonomy" of activation-related parameters/variables based on their uses in cognitive architectures (see Figure 1). At the highest level of our taxonomy, we divide activation-based functionality into three major themes: "representational," "system dynamics," and "measures of intensity or degree."

The *representational branch* is sub-divided into "associative dynamics," "availability," and "informational content." *Associative dynamics* includes "activation spreading" and the activation-based modulation of representational associations (for example, Copycat's conceptual distances). *Availability* covers the global (that is, system-wide) and bounded (that is, process or module-level) access and removal of representations. *Global*, in this context, could correspond to "forgetting" from long-term memory, and *bounded* to the eviction of representations from short-term memory. Lastly, *informational content* covers use-cases like ART's memory traces.

The *system dynamics* branch is sub-divided into "modulatory" and "scheduling" functions. *Modulatory* functions include system-wide, module-specific, or process-specific activation parameters that influence "how" operations are executed. This includes Copycat's temperature, and the notion of "arousal" from the psychological literature. *Scheduling* refers to the *deterministic* or *probabilistic* initiation of events or processes, based on activation, resulting in short-lived or long-lasting changes to a system's dynamics. LIDA's "triggers"[5] are examples of deterministic scheduling. Copycat's codelet "urgency" values are examples of probabilistic scheduling.

The *measures of intensity or degree* branch is intended to cover all activation parameters that serve to label an activation target as possessing some degree of an unambiguously defined property. This property should have a clear semantic interpretation. The magnitude of the activation indicates "to what extent" that target possesses the property. This covers, for example, Leabra's use of activation as a measure of the "truthiness" of a hypothesis.

Our Taxonomy Illustrated in LIDA

Activation is ubiquitous in LIDA, with activation-related variables and parameters supporting most (if not all) of its

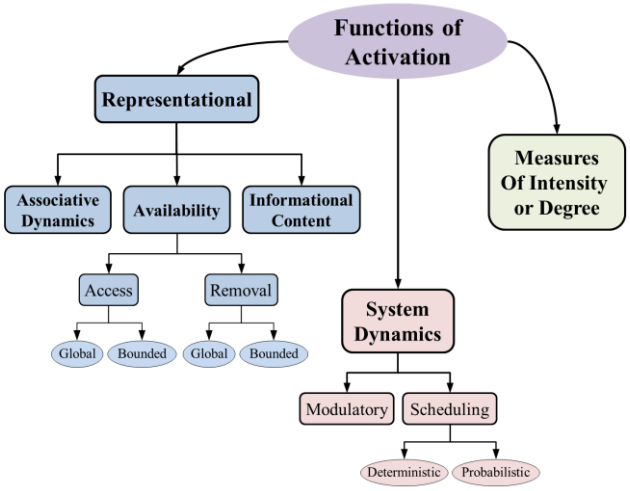


Figure 1: A "functional taxonomy" of activation. Each element specifies a category of functions that an activation variable or parameter could support.

modules, processes, and data structures. All the major themes in our functional taxonomy are present in LIDA; therefore, in this section, we use LIDA to illustrate our functional taxonomy. But first, we introduce LIDA and its activation-related concepts, so that the reader is prepared for the demonstrations that follow.

What is LIDA?

Learning Intelligent Decision[6] Agent (LIDA) (Franklin, et al., 2016) is a biologically-inspired cognitive architecture that implements, and fleshes out, significant portions of the Global Workspace Theory (GWT) of consciousness (Baars, 1988), as well as many other psychological theories (for example, Baddeley & Hitch, 1974; Barsalou, 1999; Conway, 2001; Ericsson, 1995). LIDA contains numerous short-term memory (STM) and long-term memory (LTM) modules, and special purpose processors called codelets[7]. These are depicted in Figure 2, and their functions and common acronyms are summarized in Table 1.

Cognition occurs in LIDA over a continual series of potentially overlapping "cognitive cycles," which correspond to the "action-perception cycle" referred to by many psychologists and neuroscientists (Fuster, 2004; Neisser, 1976). Each cognitive cycle is conceptually divided into "perception and understanding," attention, and "action and learning" phases. Higher-order cognitive processes such as planning, deliberation, and problem solving typically require many cognitive cycles. See (Franklin, et al., 2016) for more details.

LIDA's Activation Concepts

LIDA has historically classified its activation parameters as either "base-level activations," "current activations," or

[5] LIDA's triggers are explained in more detail later in the section entitled "Triggers."

[6] For historical reasons, this word was previously "distribution". It was later changed.

[7] This terminology was inspired by Copycat's codelets.

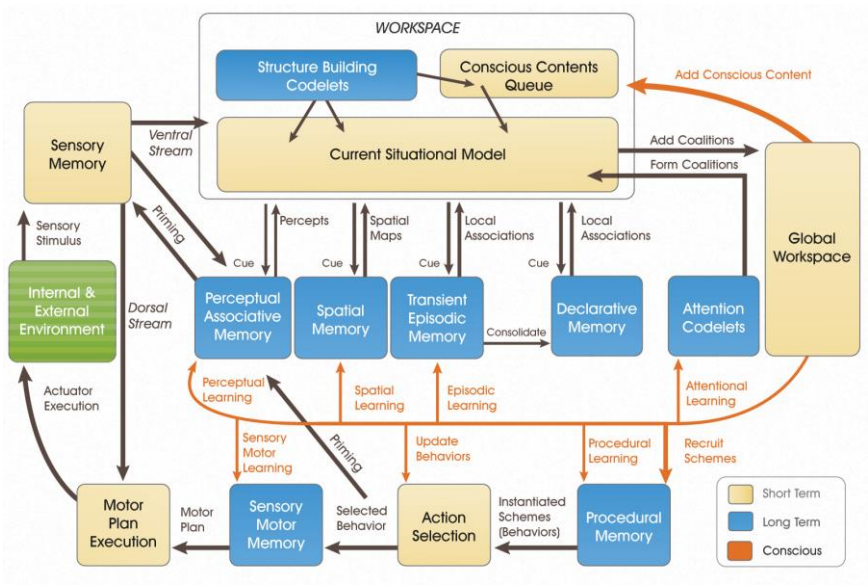


Figure 2: The LIDA cognitive cycle diagram.

simply "activations" (or "total activations"). *Base-level activation*[8] is used to describe activations with relatively slow decay rates that have activation sources based on content in the global "conscious" broadcast. These activations support "selectionist learning" (Edelman, 1987), and are the basis for the removal (forgetting) of long-term memory representations and processes. *Current activation* refers to parameters with relatively rapid decay rates that (generally) reflect transitory, module-specific notions of current "relevance." And, *activation* (or *total activation*) is used to describe all other activation parameters. Many of these general-purpose activation parameters use base-level and current activations as activation sources; however, this is not always the case.

Taxonomic Examples in LIDA

In this section, we will illustrate our taxonomy using activation-related examples from LIDA. These examples do not represent an exhaustive account of LIDA's activations; however, they should be sufficient to give the reader a taste of LIDA's major activation themes, and build an intuition for how our taxonomy could be applied in practice. Following each sub-section, we summarize the taxonomic themes that were covered.

Low-Level Feature Detectors and SM Representations. Modality-specific, low-level features detectors in Sensory Memory (SM) are activated in response to incoming sensory stimuli (from an agent's sensors). The *patterns of activation* generated in these feature detectors serve as sensory representations, corresponding to the incoming stimuli, that can be later incorporated into knowledge representations in the Workspace and long-term memory modules.

Taxonomic Themes Illustrated:
1. **Representational → Informational Content**
   [*sensory representations as "activation patterns"*]

The Activation and Instantiation of Percepts. SM uses its sensory representations to activate (that is, update the "current activations" of) feature detectors in Perceptual Associative Memory (PAM). Activation then spreads, over "activation links," to linked PAM nodes. A PAM node's activation is based on the sum of its base-level and current activations. PAM nodes with activations greater than a fixed threshold are instantiated into the Current Situational Model (CSM) as percepts, making them available to codelets and cueing processes.

Taxonomic Themes Illustrated:
1. **Representational → Associative Dynamics**
   [*spreading activation in PAM*]
2. **Representational → Availability → Access → Bounded**
   [*percepts to CSM*]

Attention Codelets and Coalitions. Preconscious content in the CSM (including percepts and other LTM representations) retain their activations after instantiation (though they subsequently decay). Some attention codelets (ACs) (for example, the "default attention codelet" described in (Franklin, et al., 2016)) use the activation associated with preconscious content to determine their "level of interest" in those representations. When an AC is "sufficiently interested" in a representation, it will take it to a coalition forming process, which may create a *coalition* containing that content. A coalition's activation is based on the activation of its content, and the base-level activation of the AC advocating for that content (among other things).

[8] LIDA's base-level activation is roughly (conceptually) analogous to ACT-R's concept of base-level activation, but its meaning is far more varied and module specific. It also has a very different activation source, which is based on LIDA's conscious broadcasts.

Table 1: Descriptions of LIDA's short-term memory (STM) and long-term memory (LTM) modules, and codelets.

Module / Process	Description
ACTION SELECTION (AS)	STM module supporting the selection of behaviors for execution by the SMS.
ATTENTION CODELETS (ACS)	Specialized processors that monitor the CSM for content of interest based on their own specific concerns, such as importance, urgency, novelty, etc. If such content is found, the codelet takes it to a coalition forming process, which may create a coalition that includes that codelet and the content it promotes.
CONSCIOUS CONTENTS QUEUE (CCQ)	STM submodule of the Workspace that contains recent conscious broadcasts.
CURRENT SITUATIONAL MODEL (CSM)	STM submodule of the Workspace that represents an agent's (preconscious) interpretation of its current situation.
GLOBAL WORKSPACE (GW)	STM module that directs a winner-take-all competition among *coalitions*, and broadcasts the content of the winning coalition in the global (conscious) broadcast.
MOTOR PLAN EXECUTION (MPE)	See SMS.
PERCEPTUAL ASSOCIATIVE MEMORY (PAM)	LTM module that supports LIDA's ability to recognize objects, events, entities, concepts, etc., and the relationships between them. The most activated representations in PAM are instantiated into the CSM as *percepts* after being activated by incoming sensory content (or cueing).
PROCEDURAL MEMORY (PM)	LTM module containing representations called *schemes* that each encode a context, action, and expected result. When schemes are instantiated (that is, when their free variables are bound to specific values based on the contents of a conscious broadcast) they are referred to as (candidate) *behaviors*.
SENSORY MEMORY (SM)	STM module that encodes modality-specific sensory content (from the environment) as the activation of low-level features detectors. These, in turn, activate perceptual representations in PAM. SM also sends sensory representations, based on the activation of its low-level feature detectors, to the CSM.
SENSORY MOTOR MEMORY (SMM)	See SMS.
SENSORY MOTOR SYSTEM (SMS)	Composed of two modules: Sensory Motor Memory and Motor Plan Execution. The SMS selects and instantiates motor plan templates from SMM into concrete motor plans, and sends them to the Motor Plan Execution module for execution.
STRUCTURE BUILDING CODELETS (SBCS)	Specialized processors that create or modify content in the CSM in support of "preconscious thought" and situational understanding.
WORKSPACE	STM module supporting preconscious, situational understanding. At any given moment it may contain cued long-term memories, percepts, sensory content (both real and simulated), transient representations created by structure building codelets. It contains two submodules—the CSM and CCQ.

Coalitions compete in a winner-take-all competition in the GW, *based entirely on the coalitions' activations*. The winning coalition's content is broadcast, making it globally accessible to all modules and processes.

Taxonomic Themes Illustrated:

1. **Representational → Availability → Access → Bounded** [*CSM to coalition forming process via default AC*]
2. **Representational → Availability → Access → Global** [*GW to global broadcast*]

Triggers. Competitions are *triggered* in the GW when a *single coalition* has an activation greater than an activation threshold, or a *set of coalitions* has activations *collectively* greater than a (different) threshold. LIDA's Action Selection module also features triggers that initiate competitions among its *behaviors* based on their activations. Since activation, in these cases, influences the rate at which conscious broadcasts occur, and actions are selected for execution, they are great examples of how activation can directly, and deterministically, influence a system's *dynamics* through event scheduling.

Taxonomic Themes Illustrated:

1. **System Dynamics → Scheduling → Deterministic** [*triggers*]

Strength of Global Broadcasts and Attentional Blinks. The *strength* of a global broadcast is determined by the magnitude of the winning coalition's activation, which is used to modulate base-level activation updates in LIDA's LTM modules (that is, selectionist learning), and update other activations in STM modules. If a broadcast's strength is "extremely" high, it can induce an "attentional blink" (Madl & Franklin, 2012); that is, a brief "refractory period" that affects all ACs, from which it gradually recovers. During the refractory period, coalitions receive less activation when they are formed; therefore, conscious broadcasts are more likely to be triggered based on the elapsed time since the last broadcast, than the coalitions' activations.

Taxonomic Themes Illustrated:
1. **Measures of Intensity or Degree** [*strength of conscious broadcast*]
2. **System Dynamics → Modulatory** [*attentional refractory period*]

Affective Valence and Feelings. LIDA's motivational system (McCall, Franklin, Faghihi, Snaider, & Kugele, 2020) is grounded in "feeling nodes"—PAM nodes with *affective valence*. Affective valence is a form of activation that quantifies notions of liking or disliking with respect to drives (hunger, thirst, etc.), or other interpretative aspects (sweetness, warmth, etc.) of real or imagined events.

Taxonomic Themes Illustrated:
1. **Measures of Intensity or Degree** [*feelings*]

Closing Remarks

In this paper, we've presented a lexicon of activation-related concepts, and a functional taxonomy that characterizes how activation has been historically applied in cognitive architectures. While we have made our best effort at gathering the major concepts and themes, it's likely that others remain. Similarly, the validity and usefulness of our taxonomy requires additional testing. Nevertheless, we hope that our efforts towards a common vocabulary will inspire activation-related discussions, and lead to a greater understanding of the concept as a whole.

Acknowledgements

We would like to thank Dr. Pulin Agrawal for his thoughts and guidance at the onset of this endeavor.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological review*, *111*(4), 1036.

Arrabales, R., Ledezma, A., & Sanchis, A. (2009). CERA-CRANIUM: A test bed for machine consciousness research. *International Workshop on Machine Consciousness.* Hong Kong.

Baars, B. J. (1988). *A Cognitive Theory of Consciousness.* Cambridge University Press.

Baddeley, A. D., & Hitch, G. J. (1974). Working memory. In G. A. Bower (Ed.), *The Psychology of Learning and Motivation* (pp. 47-89). New York: Academic Press.

Barsalou, L. W. (1999). Perceptual symbol systems. *Behavioral and brain sciences*, *22*(4), 577-660.

Conway, M. A. (2001). Sensory–perceptual episodic memory and its context: autobiographical memory. *Philos. Trans. R. Soc. Lond B.*, *356*, 1375–1384.

Duffy, E. (1957). The psychological significance of the concept of "arousal" or "activation". *Psychological Review*, 265-275.

Edelman, G. M. (1987). *Neural Darwinism: The theory of neuronal group selection.* Basic books.

Ericsson, K. A. (1995). Long-term working memory. *Psychological Review*, 211–245.

Franklin, S., Madl, T., Strain, S., Faghihi, U., Dong, D., Kugele, S., . . . Chen, S. (2016). A LIDA cognitive model tutorial. *Biologically Inspired Cognitive Architectures*(16), 105-130.

Fuster, J. M. (2004). Upper processing stages of the perception–action cycle. *Trends in cognitive sciences*, *8*(4), 143-145.

Grossberg, S. (1999). The link between brain learning, attention, and consciousness. *Consciousness and cognition*, *8*(1), 1-44.

Hofstadter, D. R., & Mitchell, M. (1995). The copycat project: A model of mental fluidity and analogy-making. In K. J. Holyoak, & J. Barnden (Eds.), *Advances in connectionist and neural computation theory, Vol. 2: logical connections* (pp. 205-267). Norwood N.J.: Ablex.

Just, M. A., & Varma, S. (2007). The organization of thinking: What functional brain imaging reveals about the neuroarchitecture of complex cognition. *Cognitive, Affective, & Behavioral Neuroscience*, *7*(3), 153-191.

Kokinov, B. N. (1994). The DUAL Cognitive Architecture: A Hybrid Multi-Agent Approach. In *11th European Conference on Artificial Intelligence* (pp. 203-207).

Kotseruba, I., & Tsotsos, J. K. (2018). 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, 1-78.

Laird, J. E. (2012). *The Soar cognitive architecture.* MIT press.

Madl, T., & Franklin, S. (2012). A LIDA-based model of the attentional blink. In *Proceedings of the 11th International Conference on Cognitive Modelling (ICCM 2012)* (pp. 283-288).

Maes, P. (1991). The agent network architecture (ANA). *Acm sigart bulletin*, *2*(4), 115-120.

McCall, R. J., Franklin, S., Faghihi, U., Snaider, J., & Kugele, S. (2020). Artificial Motivation for Cognitive Software Agents. *Journal of Artificial General Intelligence*, *11*(1), 38-69.

Neisser, U. (1976). *Cognition and reality: Principles and implications of cognitive psychology.* San Francisco: W.H. Freeman.

O'Reilly, R. C. (1996). *The Leabra model of neural interactions and learning in the neocortex.* Doctoral dissertation, PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Pezzulo, G. (2009). DiPRA: a layered agent architecture which integrates practical reasoning and sensorimotor schemas. *Connection Science*, *21*(4), 297-326.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386-408.

Shanahan, M. (2006). A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and cognition*, *15*(2), 433-449.

Computational Modeling of Human Social Intelligence and Communication

Jeungmin Lee (jeungminlee@kaist.ac.kr)
Department of Bio and Brain Engineering
Program of Brain and Cognitive Engineering
College of Engineering, KAIST, Daejeon, 34141, South Korea

Jerald D. Kralik* (jerald.kralik@gmail.com)
Department of Bio and Brain Engineering
College of Engineering, KAIST, Daejeon, 34141, South Korea

Jaeseung Jeong* (jsjeong@kaist.ac.kr)
Department of Bio and Brain Engineering
Program of Brain and Cognitive Engineering
College of Engineering, KAIST, Daejeon, 34141, South Korea

Abstract

The next level in understanding human social cognition is to model it comprehensively. To this end, we have been developing a framework and model that takes as input an event involving someone (focusing on who it was and what they did), and assesses the event based on whether it should change social accounting among individuals, and whether something should be done, such as communicating with others. Here, we present development of the model computationally and results generated by it as predictions to be tested empirically: e.g., more communication about those socially close to us when their actions are positive, and more about those with higher status (i.e., celebrities) when negative; and the relative merit or egregiousness of a wide range of behavior. Leveraging what is known of the human social mind and brain, our work aims to provide a comprehensive model of human social cognition.

Keywords: social cognition; theory of mind; communication; decision-making; computational model

Introduction

A true understanding of human social cognition should produce a comprehensive cognitive model that successfully explains human social behavior. Among many complexities, this includes successful real-time social interaction involving a great deal of inference such as for mind- and context-reading. Moreover, this inference ranges from more explicit, formal reasoning to what is considered commonsense or social intuition. Additionally, models of social intelligence will need to have not only a much richer understanding of peoples' minds and immediate context, but of sociality more fundamentally. That is, theoretical considerations (including evolutionary ones) and evidence across the social sciences have shown that social interaction can be construed in terms of social economics, with each individual — each agent in the multiagent world — having a certain amount of social value, and each social interaction a transaction, in which individuals spend and accrue social value among themselves (i.e., social accounting) (e.g., Cosmides, & Tooby, 1992; Dunbar, 2004; Foster, 2004; Pinker, 2008; Rosnow, 2001; Lee, Kralik, & Jeong, 2018; Lee, Kralik, & Jeong, 2019).

Thus, successful models will need to track and help maintain this social accounting if they are ever to be fully functional as human-level social agents and society members. Moreover, much of this social interaction resolves to deeper fundamental issues relating to morality, such as treating each other fairly. Finally, because of the primacy of these factors and the complexity of human societies more generally, successful social interactions even between two individuals often require a triad (or larger network) of people to communicate the information: e.g., due to the sensitivity of confronting someone directly or the inability to maintain accurate knowledge about others (e.g., what they have done, have learned) when not present, requiring updating from others (Baumeister, Zhang, & Vohs, 2004; Dunbar, 2004; Foster, 2004). Such indirect communication can be considered "gossip", which has seemed trivial and frivolous, but belies a deeper significance (Dunbar, 2004).

In sum, there is a vast amount of research findings on human social intelligence and sociality more generally; and yet a comprehensive theoretical framework and computational model of human social intelligence and communication has been lacking. Having this will not only organize and integrate what is currently known about human social cognition, but will also help clarify what is yet to be better understood. We have thus been developing such a framework (Lee, Kralik, & Jeong, 2018; Lee, Kralik, & Jeong, 2019). Our work has focused on integrating findings across the social sciences into a general framework and model, and here we present a brief description of the framework and initial development of the model computationally.

Lower-level, network-based architectures, including deep learning, provide a flexibility and generalization power not yet touchable for higher-level symbolic-based architectures; yet the latter reach a level of richness of human social intelligence and communication that, although currently too circumscribed (i.e., generally hand-crafted and brittle), the lower-level models have not yet broached. How the two shall meet we do not know. Thus, at least on the path to a complete

understanding of human social cognition in mind and brain, both approaches are necessary as researchers determine what architecture (or combination) can span the entire capacity of human social abilities (and perhaps beyond). Leveraging what is known of the human social mind and brain, we take a top-down theoretical approach beginning at the symbolic level. In what follows, we first briefly describe our overall framework, paradigm and model; we then describe how we have begun to implement the model computationally; and we then describe the results thus far generated, to be considered as a set of predictions for an experimental study on human social information communication that we have recently undertaken in our laboratory. The empirical study intends to test and potentially validate the modeling work with actual human findings; at the same time, the model provides deeper theoretical insight into human sociality, enabling for example, *a priori* predictions of our social behavior.

Framework, Test Paradigm, and Model

As seen in Figure 1, our general framework focuses on a *central problem-solving agent* who receives information about some activity of a *target* person, such as someone going to the movies, helping someone else in need, working well (or not) with others in a group task, beginning a romantic relationship, or cheating on an exam (Lee, Kralik, & Jeong, 2018; Lee, Kralik, & Jeong, 2019). The information is received from an *external source* or observed by the central agent directly, and based on this information she may or may not communicate with a *receiver* about it. The receiver is conceptualized broadly as anyone else, whether an additional person, and thus as "gossip", but also potentially communicating with the *source*, such as in further conversation, or with the *target* person him/herself. We call the event involving the target a "scenario" (e.g., "Kim was caught cheating on the final exam.") and it is to this point always based on someone doing something — and thus social information broadly construed.

In our test paradigm, we focus on three main factors of the event: who did it, i.e., the *target*, what they did, i.e., its *content*, and whether the content was positive or negative, i.e., its *valence*. For target identity, we have chosen to first test *ingroup* versus *outgroup* versus *celebrities*, since they enable examination of two critical social factors: *closeness* (of the target to the central agent), with the differences being ingroup > celebrities > outgroup; as well as *status*, with the differences being celebrities >> ingroup > outgroup (Aronson et al., 2016; Foster, 2004). With three *target* levels, two for *valence*, and eight *content* domains selected to span the space of activities the target may engage in (described below), we produced 48 different scenarios. With this comprehensive set, we sought to generate a set of predictions of how the various combinations are processed by the social mind/brain and drive social behavior and communication.

Model of Social Intelligence

Our model, then, is of the central agent's mind/brain, and how she determines what to do with the scenario information (Aronson, Wilson, Akert, & Sommers, 2016; Gazzaniga, Ivry & Mangun, 2013; Glimcher & Fehr, 2014; Rai, 2012; Kralik, 2017; 2018; Lee, Kralik, & Jeong, 2018; Lee, Kralik, & Jeong, 2019). To make this determination, she must process the scenario across a series of modules (that make up the central agent's social mind/brain).

We focus here on receiving information from an external source (versus observing the event directly). From a neuroscience perspective, the central agent must first sense and perceive the scenario information. Our intention is to ultimately build a system with natural input, such as via language or reading; here, however, we concentrate on more central cognitive components. When the central agent reads or hears a scenario such as "Kim was caught cheating on the final exam.", her mind/brain must first understand the basic concepts, which in our model occurs initially within the *Perception* module via accessing memory for general concept knowledge, generally realizing each word making up the sentence (like Alex, saved, child, etc.), but not the deeper meaning that the scenario is carrying. Further scenario processing is conducted in *Initial Cognitive Process* in which the stimulus takes on a deeper sense of identifying the target and the social domain at issue. This importantly includes the latter's corresponding *affect response* — as an *affect score* — such as 'Kim cheating on the final exam' relating to *fairness* or 'Alex saving a child from a fire' relating to *care* that is particularly self-sacrificing and heroic. To determine the affect score of the particular event or scenario, the *Initial Cognitive Process* accesses an *Affect Knowledge Base*, which represents our main emotion core of the model (Damasio, 1996; Gazzaniga, Ivry & Mangun, 2013; Glimcher & Fehr, 2014; Schachter & Singer, 1962; Kralik, 2017).

More specifically, we organize the *content* of possible events into eight content domains (Table 1). Five were adopted from the well-established moral foundations — prosociality, fairness, community, respect, and purity (see Haidt, 2007) — and the remaining three — competition, social-oriented, and general social affairs — were selected to represent other important social activities (Aronson et al., 2016; Dunbar, Marriott & Duncan, 1997). The affect score of each content domain was assigned via theoretical consideration and empirical evidence for the affective/emotional intensity that the domains carry. For example, events related to *prosociality* — composed of *care* for positive valence, and *harm* for negative valence (e.g., "Alex saved a child from a fire [care]" or "Sam stabbed a person with a knife [harm]") — are expected to be more intense and therefore more emotionally provoking than other content domains such as *fairness* — broken down into *fair* and *cheating* (e.g., "Kim cheated on the final exam [cheating]" — or *social-oriented*, composed of *altruism* and *selfishness* (e.g., "Taylor donated part of his salary to a charity [altruism]") (see Foster, 2004; Haidt, 2007). We discuss more about each domain in Results.

With the *affect score* as output of the *Initial Cognitive Process* module, our model then uses it as a gate for further processing, with the score representing a problem to be

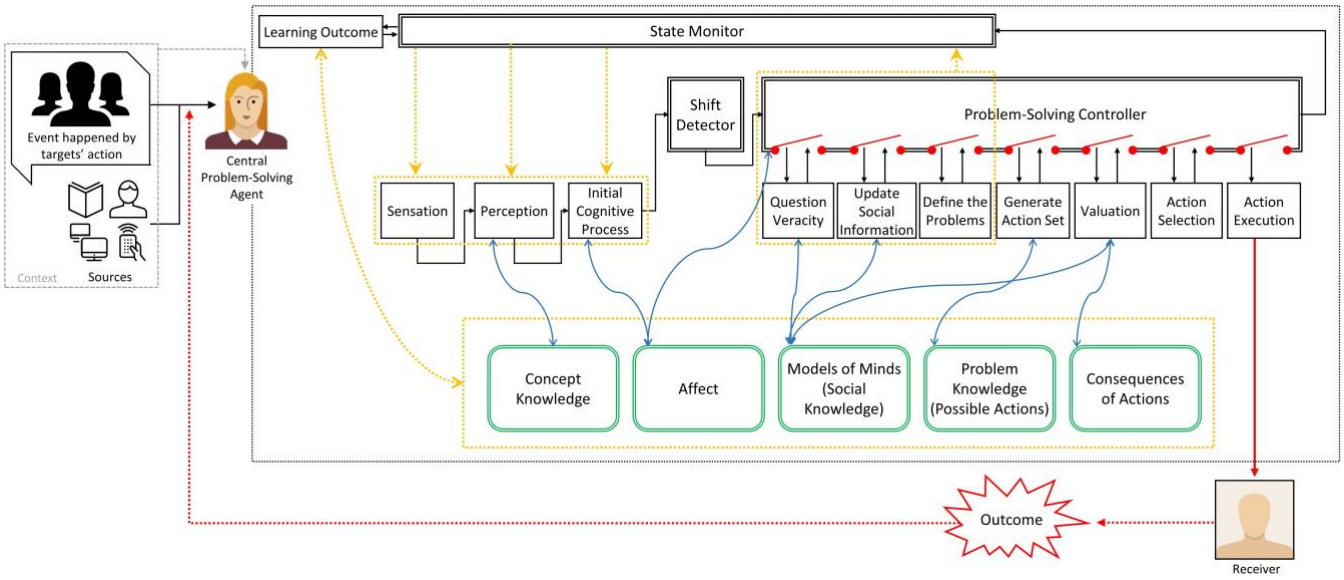


Figure 1: The complete model of the internal processes underlying social information and communication. The central agent goes through a set of internal processes (black rectangles) by accessing her relevant knowledge (green rounded rectangles).

solved or social accounting to be resolved, with the problem extent (its intensity) captured by the score value (Figure 1).

Next, a general *Problem-Solving Controller* module orchestrates problem-solving by activating a sequence of subprocesses. It is the key metacognitive process orchestrating the entire system (Kralik et al., 2018). The first subprocess assesses the likely truth of the information. Here, we assume that the source is trustworthy, and the controller then moves to *Update Social Information* where the main *Social Knowledge Base* is accessed. This knowledge base contains *models of the minds* of the people in the central agent's (multiagent) world. That is, the central agent has her own model of other people's minds (consisting of their beliefs, interests, personal traits, etc.) stored as social knowledge, accessed and used or modified when necessary (Gazzaniga, Ivry & Mangun, 2013; Glimcher & Fehr, 2014; See Lee, Kralik, & Jeong, 2018 and Lee, Kralik, & Jeong, 2019 for details).

Table 1: Eight content domains of possible social events (i.e., someone does something) along with their decomposition into positively and negatively valenced aspects, and their corresponding affect score in descending order (1-7 scale). The five domains adapted from Haidt's moral foundations are marked with superscript "m" (Haidt, 2007).

Content	*Affect Score*
Prosociality (care/harm)$_m$	7
Fairness (fair/cheating)$_m$	6
Competition (positive/negative)	5
Social-oriented (altruism/selfishness)	5
Community (loyalty/betrayal)$_m$	4
Respect (authority/subversion)$_m$	4
Purity (sanctity/degradation)$_m$	3
General social affairs (positive/negative)	1

Although in reality multiple problems are potentially in play or further introduced by the central agent's possible subsequent actions, we focus to this point on the single main problem, such as harm produced by the target's action (e.g., Sam stabbing a person with a knife) that must be resolved or in some way dealt with. Once the problem is defined, the *Problem-Solving Controller* activates the *Generate Action Set* subprocess to determine which actions to consider for the given problem (Figure 1).

Here, in our first computational development of the model, we concentrate on one main action, whether to communicate with an additional person (i.e., not the target or information source, and thus as 'gossip'), in the face of a wide range of possible social scenarios. The controller then moves to the key subprocess of *Valuation*, the central focus of our current computational model development, described comprehensively below. Upon the completion of valuation, the controller then moves to *Action Selection* and then *Action Execution*, which if the action is actual gossip, the central agent communicates with a receiver. An outcome then would occur, such as the receiver directly confronting the target, the receiver telling another receiver — that is, further gossip — or the receiver doing nothing with the information.

Valuation

We now describe the *Valuation* subprocess in detail, the central focus of our current development. *Valuation* evaluates each possible action based on the combination of potential *benefits* and *costs* of taking the action, combined with the scenario *affect score* that provides the original impetus for the problem and possible recourse to take action to resolve it.

Thus, we subtract the potential costs from benefits and multiply it by the significance of the event. More specifically, using the following equation for valuation:

$$Value_{Gossip} = A \cdot (B_{Total} - C_{Total}) \quad (1)$$

where A is the affect score of the given scenario (target person doing something), and B_{Total} and C_{Total} represent total benefits and costs of taking the given action (in this case communicating with a third-party receiver). B_{Total} and C_{Total} are each composed of multiple potential benefits and costs derived from taking a given action, i.e.,

$$B_{Total} = B_1 + B_2 + \cdots \quad (2)$$

$$C_{Total} = C_1 + C_2 + \cdots \quad (3)$$

Additionally, each individual benefit B_i or cost C_i is composed of weighting factors thus:

$$B_i = t_{Bi} \cdot v_{Bi} \cdot w_{Bi} \quad (4)$$

$$C_i = t_{Ci} \cdot v_{Ci} \cdot w_{Ci} \quad (5)$$

where t represents the relative weighting for *target* (e.g., ingroup, outgroup, or celebrity), v for scenario *valence* (i.e., positive or negative), and w for the relative influence of the individual benefits and costs.

As seen in Table 2, from the social communication literature as well as our own theoretical considerations based on an evolutionary and socio-economic perspective of how the communication about the target could ultimately benefit the central agent, we have identified five main benefits and three main costs for communicating with a third-party receiver (gossiping) (see Aronson et al., 2016; Baumeister, Zhang & Vohs, 2004; Dunbar, 2004; Foster, 2004 for reviews; Rosnow, 2001; Russell & Norvig, 2020; Lee, Kralik, & Jeong, 2018); and in the current development we have added corresponding relative weighting factors, w, t, and v, based again on this literature and our evolutionary and socio-economic theoretical considerations (Aronson et al., 2016; Baumeister, Zhang & Vohs, 2004; Dunbar, 2004; Foster, 2004; Rosnow, 2001; Russell & Norvig, 2020).

Table 2: Benefits and costs of communicating with an additional individual as receiver (i.e., gossiping). Each cell contains the weighting factors for each valuation category, based on the relative impact of each category (w), the target person's identity (t) and the valence (v) of the scenario event. See text for further description. Weights on a 1-4 scale to obtain meaningful relative values distinguishing the critical factors.

Valuation Categories	*w*	*Target (t)*			*Valence (v)*	
		Ingroup	*Outgroup*	*Celebrity*	Positive	Negative
B_1: Avoid direct contact with the target	4	4	1	1	3	4
B_2: Feedback to the gossiper from receiver	3	4	1	2.5	4	4
B_3: Update receiver's knowledge	3.5	4	1.5	2.5	4	4
B_4: Influence target's social status	4	1.5	1	4	1.5	4
B_5: Receiver influences target's behavior	3	4	1	1.5	4	4
C_1: Potential direct contact from the target	4	4	1	1	2	4
C_2: Risk of spreading wrong information	3.5	4	1	2.5	4	4
C_3: Earn bad reputation as a gossiper	3.5	4	1.5	3	2.5	4

More specifically, one potentially powerful benefit is that the central agent does not have to face the target directly (i.e., indirect communication) (B1). The advantage of such indirectness is much greater when the target person is within one's ingroup, and much less so otherwise (as celebrity or stranger), reflected in the corresponding target weights (t). The advantage is also clearer when the information is something negative about the target (e.g., Kim cheating on the final exam), although it can also be relevant for positive information (e.g., it can be uncomfortable and awkward to speak highly of someone directly to them), also reflected in the valence weights (v).

In addition, the central agent may be able to obtain more information about the scenario event circumstances or confirm the information source veracity/truthfulness by checking with a receiver (B2) (i.e., another individual in the central agent's purview), since the receiver may have more information about the target than the central agent does; and the corresponding t and v weights reflect this. At the same time, the central agent can also importantly provide the receiver with new information to update the receiver's mental model of the target (B3) (with again the relative weights reflecting this benefit based on target closeness, which also reflects the relative detail of the mental models of target individuals in the receiver's mind).

Moreover, social communication also plays an important role in society by promoting fairness in terms of social order. That is, there is an inevitable hierarchy where some individuals have higher status in terms of power, wealth, fame, etc. than others. Although *status* can refer to both macroscale hierarchy (such as nationwide or worldwide celebrities and public figures) and microscale (within a smaller social group like school, workplace, or neighborhood), we focus here on the macroscale. Social communication can potentially influence this status based on disseminating relevant information about individuals (B4) (Aronson et al., 2016; Baumeister, Zhang & Vohs, 2004; Dunbar, 2004; Foster, 2004). Furthermore, because higher status requires justification, the general public is expected to be extra judgmental and critical with those of higher status, reflected in the w, t, and v weightings.

The last benefit is also important: the possible influence of the receiver on the target to reward or "punish" them appropriately (B5). This is particularly effective (a) if the receiver is in better position to influence the target and the central agent (e.g., closer, more respected), (b) as a means to reduce a possible defensive response by the target if confronted directly, or (c) as general social pressure (i.e., reputation).

Although there are advantages to disseminating social information, there are also significant disadvantages. First, there is the possibility that the target hears of the 'gossiping' and confronts the central agent directly (C1), reflected in the weightings accordingly. The central agent also runs the risk of being wrong about the information (C2). Spreading wrong information may not only influence the target's reputation, but also actively damages the model of the target's mind in the mind of the receiver(s). The importance of having

accurate models of others is discussed in detail in our previous work (Lee, Kralik, & Jeong, 2018; Lee, Kralik, & Jeong, 2019). The last main disadvantage of information spreading is related to the traditional view of gossip as malicious behavior. By divulging information about a person (i.e., the target) absent during the conversation, the central agent may earn a bad reputation as a gossiper (C3); influenced both by closeness and valence, and thus reflected in the (*t*, *v*) weights.

We next examine our model results as a set of predictions about how the target, content, and valence of a given social act would compel someone to act on it, and in particular, to communicate with others about it.

Results

Based on the 48 scenarios produced by the combinations of *target* (ingroup, outgroup or celebrity), *content* (eight domains), and *valence* (positive or negative), and using the factors described in Tables 1 and 2 and Equations (1-5), we calculated the model's action values that reflect the likelihood of the central agent communicating to a third-party receiver (i.e., gossiping). The results are then predictions about how the human social mind processes and responds to key social information (target, content, and valence).

Figure 2 shows the model action values according to the independent effects of *valence* (Figure 2A) and *target* (Figure 2B). (Note that "total benefits > total costs" does not mean that the action will necessarily be executed, only that it increases its likelihood; and thus the key findings are the comparative values of the bar graphs.)

For *valence*, our model finds (a) a fairly comparable degree of communication (i.e., gossiping) about positive and negative events; and at the same time (b) a slightly higher amount for negative events. These predictions, especially the first of comparable amounts, are partially at odds with the prevalent view and some evidence for gossiping, in which it is believed to be predominantly negative. The model suggests that studies thus far have perhaps inordinately focused on events of negative valence (see Foster, 2004).

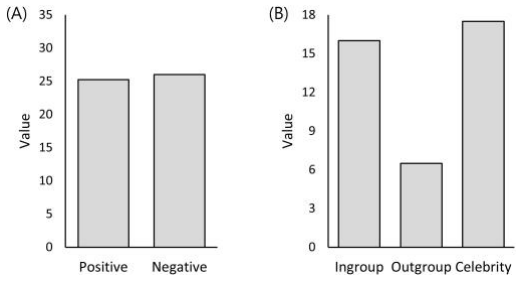


Figure 2. Predicted values of communicating social information based on its (A) valence and (B) target group.

For *target*, i.e., the person involved in the scenario event, the model predicts that celebrities will be discussed more, even more than those of one's ingroup — although again the difference between the two is not extreme (Figure 2B). Outgroup gossiping, however, is indeed predicted to be much less than the other two. The major factors underlying the target effect are *closeness* (i.e., the social distance between the target and the gossiper) and *status* of the target in terms of the larger societal hierarchy. Since outgroup is low in both status and closeness, the combination leads to the lowest amount of information spreading.

The difference between ingroup and celebrity, in contrast, is not as simple. In short, spreading ingroup and celebrity information both can have relatively high benefits (for ingroup, due to higher closeness; for celebrity, modest closeness and greater status effects; see Table 2); whereas spreading ingroup information can also lead to relatively higher costs (due to closeness, such as possible confrontation of the central agent by the target), leading to the Figure 2B result with celebrities more likely discussed even over ingroup members. Empirical evidence thus far is mixed, attesting to the need to computationally delineate the underlying factors (such as closeness, status, and the specific benefits and costs of information spreading), and to generate *a priori* predictions based on it (Foster, 2004).

Considering the potential interaction of the *target* and content *valence* of the scenario, we see a related but different predicted pattern (Figure 3A). For positively valenced scenarios (e.g., "Alex saved a child trapped in a burning building."), information involving ingroup members is predicted to be spread more than about either celebrities or outgroup members. This is due to the higher benefits yielded from ingroup information spreading, as well as the cost of ingroup information spreading decreasing dramatically and more so than in the other two target groups (e.g., no concerns about target retaliation).

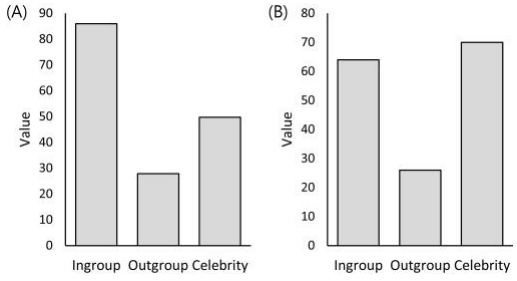


Figure 3: Predicted values of communicating information about the three target groups for scenarios of (A) positive or (B) negative valence.

For scenarios of negative valence (e.g., "Sam stabbed a person with a knife."), in contrast, the *cost* of ingroup information spreading is high; whereas the *benefit* of spreading celebrity information is high (especially due to status influence) and the risk relatively lower (Figure 3B). Therefore, with scenarios of negative valence, the model predicts that celebrity information dissemination will again be higher than for ingroup targets. Figure 4 shows more clearly the opposite patterns predicted for ingroup and celebrity targets based on the content valence.

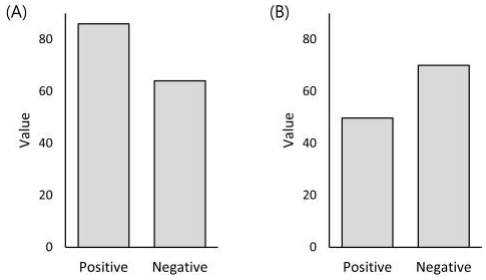


Figure 4: Predicted values of communicating information based on its valence for (A) ingroup and (B) celebrity.

Again, the empirical evidence is thus far mixed (Foster, 2004): for example, one study found that people spread more positive information about allies (friends and family) than for non-allies, including both strangers and those with high status (professors); and yet they also found more negative information spreading for *both* strangers and high-status people (McAndrew & Milenkovic, 2002). Moreover, another study found no differences in information spreading between ingroup and celebrity targets for either positive or negatively valenced events (Peng et al., 2015). Our model shows how specific factors need to be isolated to clarify the true nature of the current findings. Studies have yet to tease apart these factors sufficiently.

In sum, our computational model of social intelligence and communication has generated a number of predictions about how people evaluate the behavior of others, regarding who did it, what they did, and its general valence. These predictions include a higher degree of communication about ingroup targets when positive, and of celebrities when negative; as well as the relative merit or egregiousness of a wide range of behavior, enabling effective social interactions, generating stronger bonds among individuals, and preserving society more generally.

For model validation, we have also conducted a laboratory experiment using the same scenarios and asking participants whether they would communicate this information (e.g., Person X cheating on the final exam) to others. Thus far, our preliminary examination suggest that the model predictions are supported by the empirical results.

Discussion and Conclusions

For models of social cognition to capture human sociality they will need to include not only mind- and context-reading, but also a deeper understanding of social transactions and moral behavior. Indeed, people spend nearly 65% of their time discussing social events, suggesting that such events are especially cared about and processed (Dunbar et al., 1997). However, although a great deal is known about human social cognition, a comprehensive theoretical framework and computational model has been lacking.

To meet this challenge, we have first focused on the nuanced act of maintaining stable social behavior and societal structure by potentially communicating to others about another individual's behavior. To determine whether to do so, an individual (the central agent) must weigh its relative potential benefits and costs, which we have enumerated and quantified here. The model then makes an important theoretical contribution by producing a series of specific predictions about social communication that we are currently testing empirically (e.g., more communication about those socially close to us when their actions are positive, and more about those with higher status when negative; as well as the relative merit or egregiousness of a wide range of behavior). Indeed, in our preliminary examination of the data, the model predictions are supported.

With respect to generalizability and scale, we believe the model is poised to readily generalize and scale to larger amounts of social scenarios. For example, the critical features of all social events would be expected to resolve to *who did what*, and thus to the event's main individual(s) and the content of what happened. For target, people on first order are defined based on our relationships to them, which we have captured via ingroup, outgroup, and those of higher status. However, human understanding of others obviously goes beyond this, which in fact underscores what we believe will be the major contribution of our model: the prominence of *models of others' minds* within each agent. This component is poised to be developed substantially in the future.

Regarding content of the event, i.e., what the target did, we have organized social behavior into a comprehensive set of basic categories, ranging from deeply moral (such as *fairness*) to more everyday social activities (e.g., going to the movies). Most other types of social events are expected to fall into these categories, and thus should be readily added to the model with more detail and a mapping structure added.

Although *learning* is not yet built into the model, limiting generalizability, we have begun by adhering to evidence that suggests not only learning and cultural influences on human social cognition, but a significant underpinning of relatively hard-wired components. These include abilities such as mind-reading as well as having basic moral dimensions that appear to be universally shared and thus likely evolved, such as for *fairness* (Dunbar, Marriott, & Duncan, 1997; Dunbar, 2004; Haidt, 2007; Pinker, 2008). Learning and culture effects would then be expected to influence their relative weightings (e.g., Pinker, 2008).

In general, then, we believe a model that ultimately captures the richness of human sociality will entail both significant hard-wired components (reflecting evolutionary via genetics influences) and learned ones (reflecting culture and other environmental effects). To build such a model, we believe it is best to start with the basic more hard-wired foundations, and extend from there.

Beyond learning capabilities, an additional avenue of future development will be to determine whether the richness of human social intelligence is best captured by a symbolic-level model or ultimately resolved to a network-based one or some combination of both. In any event, we believe our framework and model help point the way forward.

References

Aronson, E., Wilson, T. D., Akert, R. M., & Sommers, S. R. (2016). *Social Psychology*. Pearson: NYC.

Baumeister, R. F., Zhang, L., & Vohs, K. D. (2004). Gossip as cultural learning. *Review of General Psychology, 8(2),* 111–121.

Cosmides, L., & Tooby, J. (1992). Cognitive adaptations for social exchange. In: J. H. Barkow, L. Cosmides, J. Tooby (Eds.), *The Adapted Mind: Evolutionary Psychology and the Generation of Culture*.

Damasio, A. R. (1996). The somatic marker hypothesis and the possible functions of the prefrontal cortex. Philosophical *Transactions of the Royal Society B: Biological Sciences, 351(1346)*, 1413–1420.

Dunbar, R. I. M., Marriott, A., & Duncan, N. D. C. (1997). Human conversational behavior. *Human Nature, 8(3),* 231–246.

Dunbar, R. I. M. (2004). Gossip in evolutionary perspective. *Review of General Psychology, 8(2),* 100–110.

Fernandes, S., Kapoor, H., & Karandikar, S. (2017). Do We Gossip for Moral Reasons? The Intersection of Moral Foundations and Gossip. *Basic and Applied Social Psychology, 39(4),* 218-230.

Foster, E.K., (2004). Research on gossip: taxonomy, methods, and future directions. *Review of General Psychology, 8*, 78–99.

Gazzaniga, M. S., Ivry, R. B., & Mangun, G. R. (2013). *Cognitive neuroscience: the biology of the mind.* WW Norton & Company.

Glimcher, P. W., & Fehr, E. (2014). *Neuroeconomics: Decision making and the brain*. Oxford: Academic Press.

Haidt, J. (2007). The new synthesis in moral psychology. *Science, 316 (5827),* 998–1002.

Kralik, J. D. (2017). Architectural design of mind & brain from an evolutionary perspective. Proc. AAAI 2017 Fall Symposium: A Standard Model of the Mind.

Kralik, J. D. (2018). Core High-Level Cognitive Abilities Derived from Hunter-Gatherer Shelter Building. In I. Juvina, J. Houpt, & C. Myers (Eds.), Proc. of the 16th International Conference on Cognitive Modeling (pp. 49-54). Madison, WI: University of Wisconsin.

Kralik, J. D., Lee, J. H., Rosenbloom, P. S., Jackson, Jr., P. C., Epstein, S. L., Romero, O. J., Sanz, R., Larue, O., Schmidtke, H. R., Lee, S. W., McGreggor, K. (2018). Metacognition for a Common Model of Cognition. *Procedia Computer Science*: 145, 730–739.

Lee, J., Kralik, J. D., & Jeong, J. (2018). A Sociocognitive-Neuroeconomic Model of Social Information Communication: To Speak Directly or To Gossip. CogSci 2018.

Lee, J., Kralik, J. D., & Jeong, J. (2019). A General Architecture for Social Intelligence in the Human Mind & Brain. AAAI Fall Symposium 2018.

Rai, T. S. Thinking in Societies and Cultures. (2012). In: Holyoak, K. J., & Morrison, R. G. (Eds.). *The Oxford Handbook of Thinking and Reasoning*. Oxford University Press.

McAndrew, F. T. & Milenkovic, M. A. (2002). Of tabloids and family secrets: *The evolutionary psychology of gossip, 32(5),* 1064-1082.

Peng, X., Li, Y., Wang, P., Mo, L., & Chen, Q. (2015) The ugly truth: negative gossip about celebrities and positive gossip about self entertain people in different ways. *Social Neuro, 10,* 320-336.

Peters, K., & Kashima, Y. (2015). Bad habit or social good? How perceptions of gossiper morality are related to gossip content. *European Journal of Social Psychology, 45,* 784-798.

Pinker, S. (2008). The Moral Instinct. *New York Times Magazine*.

Rosnow, R. L. (2001). Rumor and gossip in interpersonal interaction and beyond: A social exchange perspective. In R. M. Kowalski (Ed.), *Behaving badly: Aversive behaviors in interpersonal relationships* (pp. 203-232). Washington, DC: APA.

Russell, S. J., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach. Pearson: NJ.

Schachter, S., & Singer, J. E. (1962). Cognitive, social, and physiological determinants of emotional state. *Psychological Review, 69(5),* 379–399.

A Biologically-Inspired Neural Implementation of Affect Control Theory

Aarti Malhotra (aarti.malhotra@uwaterloo.ca)
Cheriton School of Computer Science, University of Waterloo,
200 University Ave. W, Waterloo, ON, N2L 3G1, Canada

Terrence C. Stewart (terrence.stewart@nrc-cnrc.gc.ca)
National Research Council of Canada, University of Waterloo Collaboration Centre
200 University Ave. W, Waterloo, ON, N2L 3G1, Canada

Jesse Hoey (jhoey@cs.uwaterloo.ca)
Cheriton School of Computer Science, University of Waterloo,
200 University Ave. W, Waterloo, ON, N2L 3G1, Canada

Abstract

Social interactions are a part of day-to-day life of most human beings. Affect, decision-making and behavior are central to it. With increase in adaptation of technology in our society, interaction between humans and artificially intelligent agents is also increasing. Large-scale brain-inspired neural models have been equipped with capabilities to fulfil a variety of tasks, but there has been relatively limited focus on making them capable of handling social interaction. In this paper, NeuroACT, a neural computational model and implementation of a socio-psychological theory called Affect Control Theory (ACT) is presented. This is towards building an emotionally intelligent AI agent, that can handle interactions. It takes as input a continuous affective interpretation of a perceived event, consisting of an actor, behavior and an object and generates post-event predictions of the next optimal behavior to minimize deflection. The aim is to model the role of affect guiding decision-making in AI agents, resulting in interactions that are similar to human interactions, while inhibiting some behaviors based on the social context.

Keywords: Social interaction; Affect Control Theory; Brain-inspired simulation; Nengo

Introduction

Social interaction is central to the human experience; affect and emotion plays an important role in guiding human thinking and behavior. Affect is a property of consciousness (Barrett & Satpute, 2019) and a part of every psychological phenomenon, even those that are not explicitly emotional (Hutchinson & Barrett, 2019). Social neuroscience has primarily focused on sense of self identity and how a person's mind creates a perception of another person, whereas affective neuroscience concerns mainly with brain basis of emotions (Barrett & Satpute, 2013). We present a neural model and implementation of social interaction in an AI system using a socio-psychological theory called Affect Control Theory (ACT), and combine social and affective domain perspectives, thereby dissolving their artificial boundaries. This is towards gaining deeper understanding of the neural mechanisms of the role of affect guiding decision-making. The goal here is to show that the calculations required by ACT can be implemented by spiking neurons, using anatomical structure that fits the cortex, basal ganglia, and thalamus.

Affect Control Theory (ACT) is a comprehensive social psychological theory of human interaction that emphasises feelings (affect) as a key factor (Heise, 1987; N. J. MacKinnon, 1994; Heise, 2007; N. MacKinnon & Robinson, 2014). ACT proposes that social perceptions, actions, and emotional experiences of the people are governed by a psychological need to minimize deflections between culturally shared fundamental sentiments about social situations and transient impressions resulting from the dynamic behaviors of interactants in those situations. Sentiments are said to have three aspects in an affective space, forming a three-dimensional vector comprising of Evaluation, Potency, and Activity (EPA) (Osgood, Suci, & Tannenbaum, 1957; Heise, 2007). Evaluation concerns goodness versus badness, Potency concerns powerfulness versus powerlessness, and Activity concerns liveliness versus quietness. The value of each aspect can vary in degree, in the sense that it can be greater or lesser, in either a positive or negative direction. The dimensions of EPA vector is a common cross-cultural representation of social objects, such as interactants' concepts of situations, emotions, identities and behaviors, hypothesized to be an organizing principle of human socio-emotional experience (Osgood, May, & Miron, 1975).

EPA profiles of concepts can be measured with the semantic differential, a survey technique where respondents, both males and females rate affective meanings of concepts on numerical scales. These numerically-measured sentiments are useful for mathematical analysis. EPA measurements [as noted in (Heise, 2001)] are appropriate when one is interested in affective meanings rather than denotative or logical meanings. Affective meanings correspond to sentiments - that is, the general feelings that we have about something. The EPA system is notable for being a multivariate approach to measuring affect, as compared, say, to attitude measurement which deals only with the single dimension of evaluation.

Affect control theorists have compiled datasets of a few thousand words along with average EPA ratings obtained from survey participants who are knowledgeable about their culture (Heise, 2010) [1]. For example, most English speakers agree that professors are about as nice as students (E), however more powerful (P) and less active (A). The corre-

[1] The EPA profiles in this paper are from 'Indiana 2002-4' dataset

sponding EPA profiles are {1.61, 1.58, 0.35} for professor and {1.49, 0.31, 0.75} for student. The values range by convention from -4.3 to +4.3 (Heise, 2010). In general, within-cultural agreement about EPA meanings of social concepts is high even across subgroups of society, and cultural-average EPA ratings from as little as a few dozen survey participants have shown to be extremely stable over extended periods of time (Heise, 2010).

Large-scale brain-inspired neural models like Spaun (Eliasmith et al., 2012) implement some mechanisms of cognitive processing to perform variety of tasks. A neural model called POEM (POinter EMotions) (Kajic, Schröder, Stewart, & Thagard, 2019) provides a detailed account of neurocomputational mechanisms responsible for psychological functions required for emotions. Implementing social interaction mechanisms in an AI system still remains a challenge.

In this paper, NeuroACT, a neural model of social interaction based on ACT is presented along with its implementation using Nengo, a python library for building and simulating large-scale neural models, showing the possibility of developing a human-like AI agent, which can interact with human or other agents. In this paper, the mathematical model of ACT and its non-neural implementation is first outlined. Then details about NeuroACT model, its implementation using spiking neurons and its simulation in a doctor-patient scenario and prisoner's dilemma game play are provided. Lastly, conclusion and some directions for future work are presented.

Mathematical model of ACT

In an interaction, actor is the agent who behaves (or acts) towards a target object (who can be self or some other person). An actor or an object can be specified by identity-nouns, whereas behaviors are specified by verbs.

Interaction Event Description

ACT models the formation of transient impressions from events with a minimalist grammar of the form Actor Identity-Behavior-Object Identity (*ABO*). Each of these have an EPA profile. A person's basic identity can be particularized with specifications of emotion, traits, moods, biological characteristics, statuses, or moral dispositions. In our current version of neural model, emotion is considered as the modifier. The state of being has generally more impact on the outcome impression than the identity does. [Modifier(Mod)-Identity(I) combinations were analyzed in (Smith-Lovin & Heise, 1988)]. The interaction can be written as

$$Event = [Mod_{actor}][I_{actor}][B_{actor}][Mod_{object}][I_{object}] \quad (1)$$

Note that each of these elements is an EPA profile.

Modifier-Identity Combination

The addition of attributes or adjectives that modify the identities (e.g., "good friend" or "abusive father") is calculated from the EPA values of both the identity and the modifiers. The EPA profiles of particular modifiers are symbolized as $P = \{Pe, Pp, Pa\}$, identities as $R = \{Re, Rp, Ra\}$, and the profile for identity-modifiers amalgamation as $C = \{Ce, Cp, Ca\}$. The modifier-identity profile is computed by the equation given below:

$$C = pP + rR + a \quad (2)$$

where p and r are coefficients estimated from empirical studies of the modifiers and identity, respectively and a is a constant. For example, using the affective dictionary, the EPA profile of a "stranger" is {0.02 -0.09 -0.23}, the EPA profile of "happy" is {2.92 2.43 1.96}, and that of "happy stranger" is {0.6 0.5 0.5}.

The interaction in (1) can be re-written as follows:

$$Event = [C_{actor}][B_{actor}][C_{object}] \quad (3)$$

For the rest of the paper, actor A means C_{actor} and object O means C_{object}

Deflection

According to ACT grammar, the fundamental sentiment f (represented by over-bar) is represented as follows:

$$f = \{\bar{A}_e\ \bar{A}_p\ \bar{A}_a\ \bar{B}_e\ \bar{B}_p\ \bar{B}_a\ \bar{O}_e\ \bar{O}_p\ \bar{O}_a\} \quad (4)$$

and the transient impression τ (represented by caret) evoked by an event is given by:

$$\tau = \{\hat{A}_e\ \hat{A}_p\ \hat{A}_a\ \hat{B}_e\ \hat{B}_p\ \hat{B}_a\ \hat{O}_e\ \hat{O}_p\ \hat{O}_a\} \quad (5)$$

In ACT, the weighted sum of squared Euclidean distances between fundamental sentiments and transient impressions is called total deflection D:

$$D = (f - \tau)^2 \quad (6)$$

Calculation of τ will be discussed in the next subsection. Deflection arises when impressions produced by an event differ from sentiments. Deflection that cannot be resolved produces psychological stress, which is a serious condition that can undermine one's health. Deflection is related to unlikelihood: the more deflection an event produces, the more that event seems stranger, more surprising, more unique and even inconceivable.

Consider for example, a professor who yells at a student. Most observers would agree that this professor appears considerably less nice (E), a bit less potent (P), and certainly more active (A) than the cultural average fundamentals of a professor. ACT treats the dynamics of emotional states and behaviors as continuous trajectories in affective space. Deflection minimisation is the only prescribed mechanism.

Transient impression formation The transients existing after an event can be predicted from the transients that precede the event by the equation given below:

$$\tau = Mt \quad (7)$$

M is the matrix of prediction coefficients estimated in impression-formation research, with one column for each

post-event transient being predicted. For example, Matrix M is 20 x 9, consisting of coefficients estimated from U.S male data on *ABO*. Vector t contains pre-event transients along with interaction terms that have been found to have predictive value in empirical analyses. Vector t given below is 1 x 20, hence τ is 1 x 9. [Refer to (Heise, 2007)]

$$\begin{aligned} t = \{1 \;\; \bar{A}_e \;\; \bar{A}_p \;\; \bar{A}_a \;\; \bar{B}_e \;\; \bar{B}_p \;\; \bar{B}_a \;\; \bar{O}_e \;\; \bar{O}_p \;\; \bar{O}_a \\ \bar{A}_e\bar{B}_e \;\; \bar{A}_e\bar{O}_p \;\; \bar{A}_p\bar{B}_p \;\; \bar{A}_a\bar{B}_a \;\; \bar{B}_e\bar{O}_e \\ \bar{B}_e\bar{O}_p \;\; \bar{B}_p\bar{O}_e \;\; \bar{B}_p\bar{O}_p \;\; \bar{A}_e\bar{B}_e\bar{O}_e \;\; \bar{A}_e\bar{B}_e\bar{O}_p\} \end{aligned} \tag{8}$$

To show an example of how M and t affects the calculation of τ, the following equation shows the post-event Actor's evaluation dimension estimated using the impression equations (considering non-zero values of first column of M which related to $\hat{A_e}$):

$$\begin{aligned} \hat{A_e} = -0.26 + 0.41\bar{A}_e + 0.42\bar{B}_e - 0.02\bar{B}_p - 0.10\bar{B}_a + 0.03\bar{O}_e \\ +0.06\bar{O}_p + 0.05\bar{A}_e\bar{B}_e + 0.03\bar{A}_e\bar{O}_p + 0.12\bar{B}_e\bar{O}_e \\ -0.05\bar{B}_e\bar{O}_p - 0.05\bar{B}_p\bar{O}_e + 0.03\bar{A}_e\bar{B}_e\bar{O}_e - 0.02\bar{A}_e\bar{B}_e\bar{O}_p \end{aligned} \tag{9}$$

The coefficients in the above equation indicate the factors and the degree to which they contribute towards the post-event evaluation of the actor. For example, the positive coefficient on pre-event evaluation of actor $\bar{A}_e$, means that the good actors are evaluated more positively (in E) and bad actors are evaluated more negatively (in E), with a factor of 0.41. The positive coefficient on combination terms like pre-event behavior and object evaluation $\bar{B}_e\bar{O}_e$ means that the actors are evaluated more positively (in E) if they are observed doing good things to good people, or bad things to bad people, but more negatively (in E) if they are observed doing bad things to good people or good things to bad people, with a factor of 0.12. Similarly, the other dimensions can be calculated for $\hat{A}$, $\hat{B}$ and $\hat{O}$ giving us the value of τ as mentioned in eq. (5).

Optimal Behavior

Action selection in an interaction would be based on any institutionally-appropriate, feasible, and sentiment-affirming behavior. For example, in a medical setting, there would be a doctor-patient interaction, where doctor's identity is generally considered as quite good and potent and somewhat active with an EPA profile as $\{1.90,0.69,0.05\}$, whereas a patient identity is considered a bit good, less powerful and quite weak with an EPA profile as $\{0.90,-0.69,-1.05\}$. The sentiment-affirming behavior for a doctor would be to treat, instruct etc to the patient, so that his impression is maintained as good. If he does acts of yelling, cruelty etc, his impression will be bad and will cause deflection and conflict.

An event seems more unlikely, uncanny, or unique as deflections (D) are larger. In ACT, the EPA profile for the optimal behavior is regarded as the one that minimizes the unlikeliness of an event, that is defined as below.

$$k + \sum_{i=A_e}^{O_e} w_i D_i \tag{10}$$

From eq. (10) and (6), we have

$$k + \sum_{i=A_e}^{O_e} w_i (f_i - \tau_i)^2 \tag{11}$$

where k is a constant and w stands for summation weights. Minimizing unlikeliness or maximizing normality is obtained by setting partial derivatives of the right side of the above equation to zero and solving for behavior terms, giving us the suggested optimal behavior [for details refer (Heise, 2007)].

Predicted Emotion and Identity

ACT predicts the emotion and identity of both the actor and the object post interaction, which can also affect the dynamics. In this paper only optimal behavior will be focused upon.

Non-Neural implementations

Interact A computer software tool named Interact, implements ACT's mathematical model in Java. It provides a user interface to setup the interactions and analyze the results. It has a dictionary of various datasets across six nations, ranging from 1977 to 2007, and consists of EPA profile ratings for identities, behaviors, modifiers rated by male and female raters, which is useful in cross-cultural and historical analysis. [New datasets can be found at https://research.franklin.uga.edu/act].

BayesAct BayesAct (Hoey, Schröder, & Alhothali, 2016; Schröder, Hoey, & Rogers, 2016) generalises ACT by maintaining multiple hypothesis of behaviors and identities simultaneously as a probability distribution. It uses partially observable Markov decision process (POMDP). [Some applications include (Lin et al., 2014; Jung, Hoey, Morgan, Schröder, & Wolf, 2016)].
BayesAct and Interact can be accessed at http://bayesact.ca

Neural Model

The novel contribution of this paper is to take the underlying mathematics of ACT and implement them using the spiking neurons. In particular, it is striking that the overall form of the theory maps very well onto a neural model of the cortex/basal ganglia/thalamus loop that has been previously used to model a variety of tasks (Eliasmith et al., 2012).

The core part of the algorithm that is modelled here and its relation to the neural model of the brain is shown in Figure 1. In this work, the mechanisms for maintaining and tracking the EPA values of the current situation is not modelled; rather, focus is on the calculation of deflection and hence unlikeliness, given the event perception from an object's (AI agent) perspective and time t. That is, given the EPA values of the current situation, the question is: what action should be performed by the object of the event?

This maps well onto the traditional roles of the cortex, basal ganglia, and thalamus. Neurons in the cortex (1 in Figure 1) will represent the EPA values, the connections between cortical neurons and basal ganglia neurons (2) will compute

eq. (11), the basal ganglia (3) will find the action with the largest deflection minimizing utility value, and the thalamus (4) will activate that particular action.

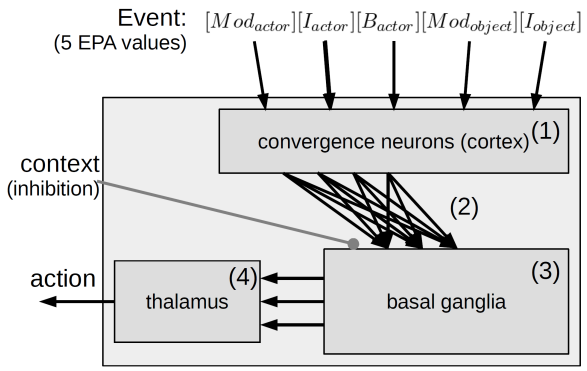


Figure 1: Neural implementation of ACT

While the overall mathematical function of this system is easy to describe and implement, it will be shown how spiking neurons can perform these operations. In particular, here Neural Engineering Framework (NEF; Eliasmith & Anderson, 2004) is used, which is a general method for finding how to connect simulated neurons so as to get the best approximation of any given algorithm. In general, the idea here is that the activity of groups of neurons can be thought of as representing vectors, and the connections between groups of neurons can be thought of as computing functions on those vectors. If we know the set of functions that we want to compute then we can perform a sequence of local optimizations (one for each set of connection weights) that will find the best approximation of the algorithm, given whatever type of neurons we want to use (including spiking and non-spiking neuron models).

For the basal ganglia and thalamus, we can make use of already-existing models of how to use the NEF to implement exactly the function that is desired here: a system that takes in a set of values from eq. (11) and determines which one is the largest utility, say U, outputting that information to the thalamus. This has been previously shown to both map on well to the anatomy of the basal ganglia and to exhibit realistic reaction times (Stewart, Choo, & Eliasmith, 2010). This system has been used in many previous models, including models of the bandit task (Stewart, Bekolay, & Eliasmith, 2012) and the large functional brain model Spaun (Eliasmith et al., 2012). The same is used here without adjusting any parameters. Also, an inhibitory "context" input that provides a large negative value for any actions that should not currently be considered.

While the basal ganglia and thalamus model take care of computing which of the action values has the largest deflection minimizing utility U (i.e. which action should be taken), this still leaves the question of how to have neurons calculate the eq. (11) values for each action, given the basic EPA values that constitute t.

Since this is simply a function, it is possible to train a neural network to approximate that function. However, the general challenge of neural networks is that if the function being approximated is too complicated, we will need a very large neural network to do this (either very deep or very broad, or both). Importantly, the networks generated using the Neural Engineering Framework have been analyzed in terms of the class of functions that they are good at approximating when using a Leaky Integrate-and-Fire neuron model with the default distribution of tuning curves (Eliasmith & Anderson, 2004). This analysis indicates that these neurons are best at approximating functions that consist of linear combinations of low-degree polynomials. Crucially, this is *exactly* the form of the calculation being done here (see eq. (9)). This means that we can use small numbers of neurons (here we use 1500) with the same parameter settings as has been used in the other biological models to approximate this function.

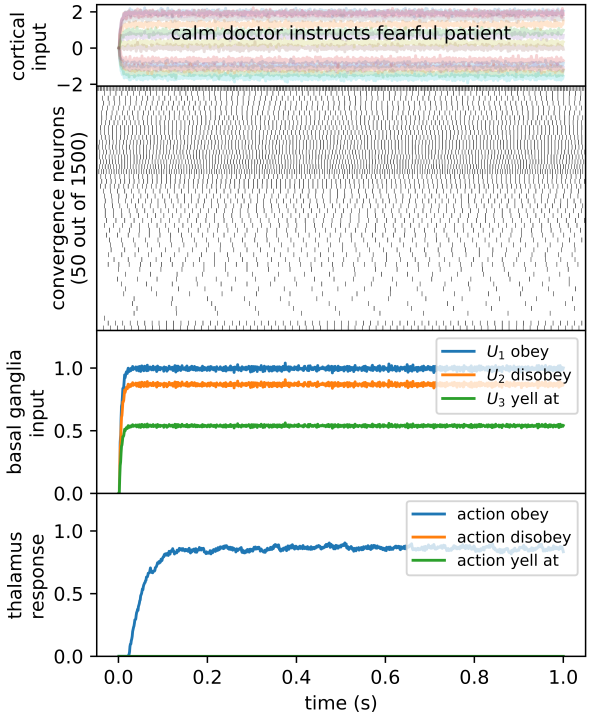


Figure 2: Example of behavior of NeuroACT

An example of the overall behavior of the resulting model is shown in Figure 2. The input is the EPA values for each of the 5 relevant terms. In this case, the situation is

$$[calm][doctor][instructs][fearful][patient]$$

and the corresponding input EPA values are [1.97 1.32 -1.4][1.9 0.69 0.05][1.85 1.65 0.3][-1.64 -0.94 -1.15][0.9 -0.69 -1.05]. These values are fed into the convergence neurons. These connections are completely random, meaning that any particular input will produce some random pattern of neural activity that is unique to that input. From that activity, the connection weights from the convergence neurons to the

basal ganglia compute the eq. (11) function for all of the different actions in parallel. For simplicity, here we only plot three of those actions: 'obey', 'disobey' and 'yell at'. Finally, the basal ganglia model finds the largest of these activity values (i.e. 'obey') and directs that result to the thalamus, so the object of the event, which is the patient in this case, can perform for better interaction. This is also the optimal behavior according to the mathematical model.

Simulation

To simulate NeuroACT model for social interaction involving affect, decision-making and behavior, a single play of prisoners dilemma game scenario was used. Of the two players involved; one represents a simulated human player agent (Actor) and the other represents NeuroACT AI agent (Object). In the play round, each player can decide to either give two coins to the other player (cooperation strategy) or take one coin (defection strategy) from a common pile. Players can maximize their returns by defecting while their partner cooperated, and although the Nash equilibrium is mutual defection, the players can jointly maximize their scores through mutual cooperation.

In the simulation scenarios, the AI agent perceives the emotional state, identity and behavior of the human agent, and outputs the optimal behavior it would choose ('give' or 'take') based on the ACT prescription of deflection-minimization. The decision-making dynamics over the time scale are tested, such that if the perceived emotion of the human agent changes during the play round, the AI agent changes its strategy as well. For simplicity, the identity of both the players was kept as 'stranger'. EPA profiles used for identity, modifiers and behaviors are as below:

$$[happy] : [2.92, 2.43, 1.96]$$

$$[angry] : [-1.45, -0.30, 1.13]$$

$$[stranger] : [0.02, -0.09, -0.23]$$

$$[gives\ to] : [1.60, 1.47, 1.55]$$

$$[takes\ from] : [-1.40, 1.62, 1.50]$$

Inhibition: The dictionary consists of 500 behaviors, out of which 498 are inhibited in this case due to the game context. If there was no mechanism of inhibitory neurons, AI agent would have selected a deflection-minimizing behavior out of 500 options, but in our case, it selects between 'give' or 'take' behavior only and others get inhibited.

To demonstrate the behavior of the model and show its ability to use neurons to perform similar calculations as found in the standard Affect Control Theory, we provide cortical input of 5 sets of EPA values representing a particular situation. Since neurons require time to respond, we hold this input constant for 0.5 seconds and then present a new situation. In particular, we manually adjust the recognized emotion from 'happy' to 'angry', as this causes ACT to produce a different action. It should be noted that, in this example, the 'object' is meant to correspond to the NeuroACT AI agent itself.

Scenario 1: Human agent cooperates with AI agent

Perception at time $t \leq 0.5$:

$$[happy][stranger][\textbf{gives to}][happy][stranger]$$

Perception at time $t > 0.5$:

$$[\textbf{angry}][stranger][\textbf{gives to}][happy][stranger]$$

Scenario 2: Human agent defects with AI agent

Perception at time $t \leq 0.5$:

$$[happy][stranger][\textbf{takes from}][happy][stranger]$$

Perception at time $t > 0.5$:

$$[\textbf{angry}][stranger][\textbf{takes from}][happy][stranger]$$

Results

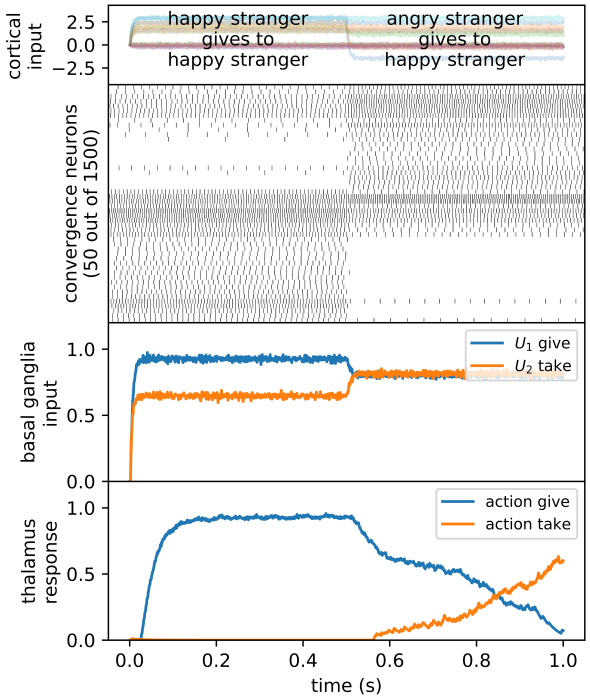


Figure 3: Human agent cooperates with AI agent

Results for the simulation runs for Scenarios 1 and 2 are shown in Fig 3 and 4 respectively. In both scenarios, the resultant behavior changes from 'give' to 'take' on perceiving the emotion of the human agent that changes from 'happy' to 'angry', given the affective dynamics. In scenario 1 (Fig 3), the change in behavior seems slower and more deliberate than

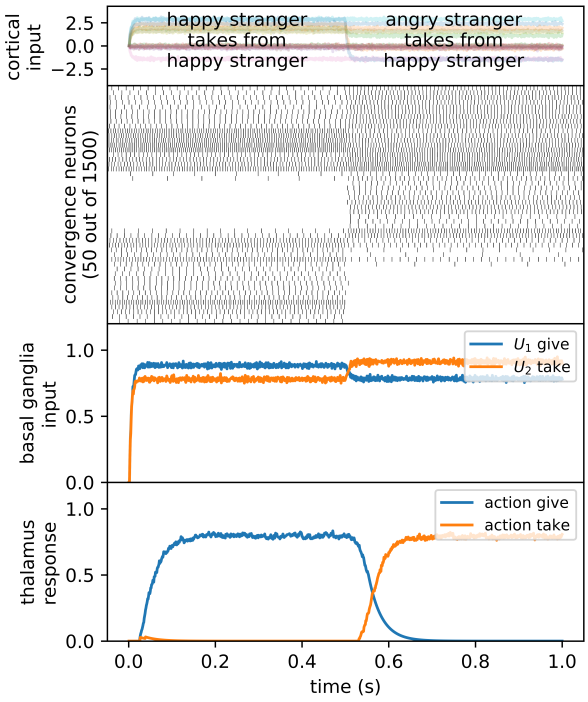


Figure 4: Human agent defects with AI agent

in scenario 2 (Fig 4), where the change is faster and somewhat automatic. This may be due to actor's behavior being more positive in scenario 1 as compared to scenario 2.

NeuroACT shows how affect influences decision-making and behavior. The behavior chosen by the model matches with its non-neural counterpart in choosing the optimal behavior as prescribed by ACT. The ability of the neural model to handle time dimension is important for the temporal order of information processing similar to human brain circuit (Gupta & Merchant, 2017).

Conclusion and Future Work

Social interaction is a challenging area to replicate in brain simulations. NeuroACT is a novel contribution implementing affective social interaction in spiking neurons. It is a generalized and extensible neural model of ACT, capable of providing an AI agent with the ability to interact with the other AI agents or humans. The input is an interaction perception and output is an optimal behavior selection. This is a step towards making emotionally intelligent AI agents.

A specific doctor-patient interaction is tested for the model. Simulation of a single play in Prisoner's dilemma game is provided. This can be iterated as well, taking into account that in the next round of play, the actor and the object change. NeuroACT can be used to model any other interaction. Future enhancement can include settings for additional context, such as location. The model can be expanded using similar methods to predict the emotion and generate re-identification of the actor and the object post-interaction. This system can be enhanced by incorporating some sensorimotor signals to integrate with physical robots.

Some other improvements could be considered involving a working memory component for the agent to utilize experience from the previous interactions. The input to the model is a generic input, that can incorporate visual, textual, or auditory forms, as all would eventually translate into verbal concepts. Advances in neuroimaging techniques like hyperscanning to study the inter-brain synchronisation (Liu et al., 2018) in social interaction may give more insight into the neural mechanisms at play.

References

Barrett, L. F., & Satpute, A. B. (2013). Large-scale brain networks in affective and social neuroscience: towards an integrative functional architecture of the brain. *Current Opinion in Neurobiology*, *23*, 361-372.

Barrett, L. F., & Satpute, A. B. (2019). Historical pitfalls and new directions in the neuroscience of emotion. *Neuroscience Letters*, *693*, 9-18.

Eliasmith, C., & Anderson, C. H. (2004). Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems. *IEEE Transactions on Neural Networks*, *15*, 528-529.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A Large-Scale Model of the Functioning Brain. *Science*, *338*(6111), 1202–1205.

Gupta, D. S., & Merchant, H. (2017). Understanding the Role of the Time Dimension in the Brain Information Processing. *Frontiers in Psychology*, *8*, 240.

Heise, D. R. (1987). Affect control theory: Concepts and model. *The Journal of Mathematical Sociology*, *13*(1-2), 1–33.

Heise, D. R. (2001). Project Magellan: Collecting Cross-cultural Affective Meanings Via The Internet..

Heise, D. R. (2007). *Expressive order: Confirming sentiments in social actions*. Springer US.

Heise, D. R. (2010). *Surveying Cultures: Discovering Shared Conceptions and Sentiments*. John Wiley & Sons, Inc.

Hoey, J., Schröder, T., & Alhothali, A. (2016). Affect Control Processes: Intelligent Affective Interaction using a Partially Observable Markov Decision Process. *Artificial Intelligence*, *230*, 134–172.

Hutchinson, J. B., & Barrett, L. F. (2019). The Power of Predictions: An Emerging Paradigm for Psychological Research. *Current Directions in Psychological Science*, *28*, 280 - 291.

Jung, J. D. A., Hoey, J., Morgan, J. H., Schröder, T., & Wolf, I. (2016). Grounding Social Interaction with Affective Intelligence. In *Canadian Conference on AI*.

Kajic, I., Schröder, T., Stewart, T. C., & Thagard, P. (2019). The semantic pointer theory of emotion: Integrating physiology, appraisal, and construction. *Cognitive Systems Research*, *58*, 35-53.

Lin, L., Czarnuch, S., Malhotra, A., Yu, L., Schröder, T., & Hoey, J. (2014). Affectively Aligned Cognitive Assistance Using Bayesian Affect Control Theory. In *IWAAL*.

Liu, D., Liu, S. Z., Liu, X., Zhang, C. Q., Li, A., Jin, C., … Zhang, X. (2018). Interactive Brain Activity: Review and Progress on EEG-Based Hyperscanning in Social Interactions. *Frontiers in Psychology*, *9*, 1862.

MacKinnon, N., & Robinson, D. (2014). Back to the Future: 25 Years of Research in Affect Control Theory. *Advances in Group Processes*, *31*, 139–173.

MacKinnon, N. J. (1994). *Symbolic interactionism as affect control.* Suny Press. (Pages: xvi, 245)

Osgood, C. E., May, W. H., & Miron, M. S. (1975). *Cross-cultural universals of affective meaning.* University of Illinois Press. (Pages: xix, 486)

Osgood, C. E., Suci, G. J., & Tannenbaum, P. H. (1957). *The measurement of meaning*. University of Illinois press.

Schröder, T., Hoey, J., & Rogers, K. B. (2016). Modeling dynamic identities and uncertainty in social interactions: Bayesian affect control theory. *American Sociological Review*, *81*(4), 828–855.

Smith-Lovin, L., & Heise, D. R. (1988). *Analyzing social interaction: Advances in affect control theory*. Taylor & Francis.

Stewart, T. C., Bekolay, T., & Eliasmith, C. (2012). Learning to select actions with spiking neurons in the basal ganglia. *Frontiers in Decision Neuroscience*, *6*, 2.

Stewart, T. C., Choo, X., & Eliasmith, C. (2010). Dynamic Behaviour of a Spiking Model of Action Selection in the Basal Ganglia. In *Proceedings of the 10th International Conference on Cognitive Modeling* (pp. 235–40).

Modelling Human Information Processing Limitations in Learning Tasks with Reinforcement Learning

Tyler Malloy (mallot@rpi.edu)

Chris R. Sims (simsc3@rpi.edu)
Department of Cognitive Science
Rensselaer Polytechnic Institute
110 8th St, Troy, NY 12180

Keywords: Reinforcement Learning; Information Theory; Bounded Rationality; Rational Inattention

Introduction

In behavioral economics, 'rational inattention' (C. A. Sims, 2010) has been proposed as a theory of human decision-making subject to information processing limitations. This theory hypothesizes that decision-makers act so as to optimize a trade-off between the utility of their behavior, and the information processing effort required to reach a good decision. Shannon information has been proposed as a means of quantifying this information processing cost. However, existing models in the rational inattention framework do not account for the learning dynamics that underlie human decision-making. In order to incorporate the impact of cognitive limitations on learning, we extend the traditional reinforcement learning objective to incorporate a bound on the Shannon information of the learned policy (see also Lerch & Sims, 2019). Using experimental data from a previously-studied learning paradigm (Niv et al., 2015), we show that our method can be used to represent differences in participants' performance as resulting in part from utilizing different capacities for storing and processing information.

Rational Inattention

According to theories of rational inattention, human decision-makers seek to maximize the following objective (Jung, Kim, Matějka, & Sims, 2019):

$$\max \mathrm{E}[U(X,Y)] - \lambda I(X,Y), \tag{1}$$

where $U(X,Y)$ describes the utility of choice Y in state X, and $I(X,Y)$ represents the mutual information between the state X and the action Y.

The result of altering the traditional expected utility maximization with a regularization term based on mutual information is a constraint on the information-theoretic complexity of the decision-makers' behavior. This limitation is proportional to the scale of the parameter λ; as λ increases, simpler policies will be preferred over increased expected utility. In the extreme, a decision-maker would act randomly or else choose the same action regardless of his or her state (ignoring all information from the environment).

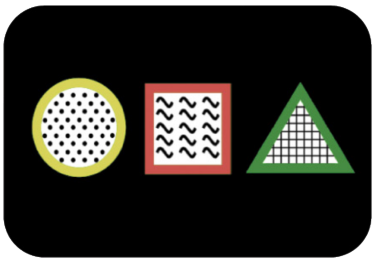


Figure 1: Example of stimuli used in the Niv et al. paradigm. Each of the 9 features was randomly assigned to one of the three possible objects, with no feature present more than once in the same stimulus.

Feature Reinforcement Learning

The specific reinforcement learning algorithm we are interested in extending is a variant of Q-learning defined in (Niv et al., 2015) called Feature Reinforcement Learning (FRL). The algorithm defines the value of an option $V(S)$ in a contextual n-armed bandit learning task to be the sum of the values of the features that make up that option:

$$V(S) = \sum_{f \in S} W(f), \tag{2}$$

where the weights of each feature are updated based on the selection that was made by the participant and the reward that was observed as follows:

$$W^{\text{new}}(f) = W^{\text{old}}(f) + \eta[R_t - V(S_{\text{chosen}})] \; \forall f \in S_{\text{chosen}}. \tag{3}$$

FRL was developed to explain human learning performance in domains with high-dimensionality. In their experiment, participants were presented with stimuli varying in color, shape, and texture. Each feature dimension had three possible feature values (for example, stimuli could be red, green, or yellow). The task for participants was to learn which of the nine possible features leads to the highest probability of reward (Figure 1), changing roughly every 20 episodes.

The results shown in (Niv et al., 2015) indicate that it is possible to achieve high predictive accuracy on the selections made by participants using the standard FRL model. In the following section we show that greater predictive accuracy can be achieved by determining the capacity for storing

and processing information that is used by each of the participants, and modelling their resulting behaviour with the capacity-limited FRL method.

Capacity-Limited FRL

Applying the learning objective defined in (1) onto the domain of reinforcement learning results in an algorithm that allows us to define a capacity for the amount of information that is used to represent our agent's policy. The two additional hyper-parameters are the capacity-limit C, which is determined for each participant individually using the same method as described in (Niv et al., 2015), as well as the feature weight adjustment learning rate $\alpha = 1e-3$ for all participants.

Algorithm 1: Capacity-Limited FRL

Initialize: Feature weights $W(f) = \bar{0}$
Initialize: Hyper-parameters: α, β, η, δ, C
for *each participant selection S* **do**
  Predict choice with probability distribution $\pi(A|S)$
  for *each feature f in selection S* **do**
    $W^{\text{new}}(f) = W^{\text{old}}(f) + \eta[R_t - V(S_{\text{chosen}})]$
  for *each feature f not in selection S* **do**
    $W^{\text{new}}(f) = (1-\delta)W^{\text{old}}(f)\ \forall f \notin S_{\text{chosen}}$
  while $I(\pi(a|s)) > C$ **do**
    for *each f in W(f)* **do**
      $f = f - \alpha(f - \sum_{f \in F} W(f)/|W(f)|)$

The constraint on the amount of information used to represent performance is determined by the magnitude of the capacity parameter C, which performs the same function as the parameter λ in Eq (1). Decreasing the value of C results in a more and more strict limitation on the amount of information that is used by the model to represent the performance of the participant. The algorithm iteratively updates the RL Q-table to decrease the mutual information until it is below the bound. In the next section, we fit this parameter to each of the individual participants performance in the contextual n-armed bandit learning environment. This algorithm demonstrates that the mutual information regularized expected utility maximization approach that is described in Eq (1) is applicable into the domain of human reinforcement learning.

Results

The original experiment design described in (Niv et al., 2015) includes 2 different speed trials, fast (500ms) and slow (1.5s) response times, with the slow response times used during trials to allow for a fMRI scanner enough time to capture data for a separate analysis that is not discussed further. Hyper-parameters were originally fit by minimizing negative log posterior individually for both the slow and fast trials. However, one potential benefit of the capacity-limited approach is that the information capacity parameter C could be the same across different tasks for the same participant, as long as factors such as motivation and attention remain consistent enough across the different tasks. To support this, we instead fit both models to the entire data set for individual participants using the Python Scipy minimization package, and compare the performance of the FRL and CLRL methods. These results indicate that it is possible to determine the information capacity that is used by a participant in a learning task, even across tasks with slightly different cognitive requirements such as the different time constraints shown here.

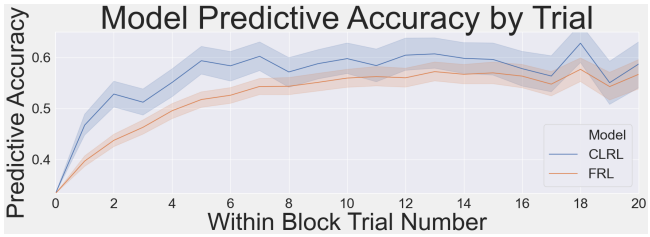


Figure 2: Mean predictive accuracy of CLRL and FRL models based on parameters fit to minimize negative log loss across both fast (500ms) and slow (1.5s) response times. Error bars represent 99% confidence intervals.

The high predictive accuracy of the CLRL model when fit to the entire data set demonstrates a similarity of participant's information processing capacities across different tasks. Although the individual sources of these capacities can be varied, from attention and motivation to differences in cognitive abilities, this model determines the amount of information required to represent participants' learned behaviour. This difference represents one possible explanation for less than optimal performance on learning and decision making tasks that is observed with human participants. By connecting the information-constrained maximum utility with reinforcement learning, this algorithm expands the application into learning tasks. In developing this algorithm, we further support the conceptualization of rational decision makers as Shannon information channels with a limited capacity for storing and processing information that is efficiently allocated to maximize reward when learning and making decisions.

References

Jung, J., Kim, J. H., Matějka, F., & Sims, C. A. (2019). Discrete actions in information-constrained decision problems. *The Review of Economic Studies*, *86*(6), 2643–2667.

Lerch, R. A., & Sims, C. R. (2019). Rate-distortion theory and computationally rational reinforcement learning. *Proceedings of Reinforcement Learning and Decision Making (RLDM) 2019*, 7–10.

Niv, Y., Daniel, R., Geana, A., Gershman, S. J., Leong, Y. C., Radulescu, A., & Wilson, R. C. (2015). Reinforcement Learning in Multidimensional Environments Relies on Attention Mechanisms. *Journal of Neurosci*, *35*(21), 8145–8157. doi: 10.1523/JNEUROSCI.2978-14.2015

Sims, C. A. (2010). Rational inattention and monetary economics. In *Handbook of monetary economics* (Vol. 3, pp. 155–181). Elsevier.

Neurally-informed modelling of static and dynamic decision biases

L. Alexandra Martinez-Rodriguez (l.martinezrodriguez@ucdconnect.ie)
School of Electrical and Electronic Engineering
University College Dublin, Dublin 4, Ireland

Elaine A. Corbett (corbetel@tcd.ie)
Trinity College Institute of Neuroscience and School of Psychology
Trinity College Dublin, Dublin 2, Ireland

Redmond G. O'Connell (reoconne@tcd.ie)
Trinity College Institute of Neuroscience and School of Psychology
Trinity College Dublin, Dublin 2, Ireland

Simon P. Kelly (simon.kelly@ucd.ie)
School of Electrical and Electronic Engineering
University College Dublin, Dublin 4, Ireland

Abstract

Different accounts have been developed to explain the mechanisms underlying value biases during perceptual decision-making, within the model framework of bounded accumulation. The starting point bias account suggests a shift in the starting point of evidence accumulation, in the direction of the more valuable alternative. The drift rate bias account suggests that the mean rate of accumulation is steepened for the more valuable alternative. While most studies have supported a starting point bias (SPB) approach, recent work (Afacan-Seref, Steinemann, Blangero, & Kelly, 2018) suggests that drift rate biases (DRB) may also be applied in certain circumstances. Here, we used human EEG signatures of competitive motor preparation to construct a cognitive decision model that can explain the biasing mechanisms through which participants perform a value-biased orientation discrimination task under a strict deadline. Motor preparation dynamics showed signs of a value bias that emerged prior to evidence onset and increased steadily with time. Accordingly, we constructed a model that included an anticipatory dynamic urgency signal towards the High Value alternative. This model provided a better fit to behaviour than models with either a starting point or a drift rate bias but no anticipatory dynamics. These results point to a role for value-modulated, anticipatory motor preparation in fast-paced decision-making tasks, and suggest a unitary mechanism that can generate both static (starting point) and dynamic (drift rate) biases at the same time.

Keywords: value-biased decisions, urgency, decision making.

Introduction

Simple perceptual decisions can be divided into three key processing stages: sensory encoding (the representation of sensory information in the brain), decision formation and motor execution. Under no time constraints, sensory encoding and decision formation can typically be completed well in advance of motor execution. However, in many real-life situations, such as when playing football or driving at speed in traffic, the brain has a limited time to integrate sensory information before a response must be produced. In such situations, motor processes must evolve swiftly and prioritise the action associated with higher value in order to maximise overall expected rewards, at the cost of greater uncertainty about the correct choice. The way in which the brain implements such prioritisation remains unclear.

For decades, sensorimotor decisions (where perception is translated into overt action) have been studied using Bounded Evidence Accumulation models. In this framework, sensory evidence is integrated over time into a "decision variable", that produces an action upon reaching a threshold. Different model variants have been devised to explain why choices tend to be biased towards the more valuable option. The most prominent of them incorporate static biases that do not change in time, such as the Starting Point Bias (SPB) model, where the starting point of the decision variable is shifted towards the higher value option. The main alternative to this is a Drift Rate Bias (DRB), where the mean rate of accumulation is biased by value, and results in an increasing displacement of the decision variable with time. In principle, one way that a DRB can arise is from an enhancement of representations of higher-value alternatives at the sensory level or in the weighting of their readout, because stronger sensory evidence would lead to a steeper build-up of its integral. Another way is through the addition of a dynamic bias signal at the motor level, known as urgency. Urgency is an evidence-independent component of decision variable buildup that contributes to bringing the neural activity closer to a given neural threshold even in the absence of informative sensory evidence (Hanks, Kiani, & Shadlen, 2014). The decision-making literature reflects a preference for static (starting point) biases, because the models that incorporate them usually offer an excellent quantitative fit to response time (RT) distributions across many psychophysical tasks (Ratcliff & McKoon, 2008; Hawkins, Forstmann, Wagenmakers, Ratcliff, & Brown, 2015).

Although not favoured in cognitive model comparisons, other evidence suggests the plausibility of drift rate biases. For example, studies have shown that sensory cortical representations of stimuli are altered through their association with reward, in a manner resembling effects of spatial or feature-based attention (Serences & Saproo, 2010; Stanisor, Van Der Togt, Pennartz, & Roelfsema, 2013). There has also been em-

pirical evidence for the operation of dynamic urgency, which has been observed in the firing of neurons associated with motor preparation in saccade-decision tasks in monkeys (Hanks, Mazurek, Kiani, Hopp, & Shadlen, 2011). In that study, the authors used a motion discrimination task, where they manipulated stimulus reliability and the prior probability of motion direction. Their results showed that a model incorporating a dynamic bias signal that adds to cumulative evidence and increases as a function of decision time - effectively generating a DRB - provided a significantly better account of the data than a static signal implemented as a SPB.

In a recent human EEG study of rapid color discrimination under a very strict deadline, it was shown that a DRB model in which a constant value-bias is added to an increasing drift rate outperformed both standard SPB and DRB models (Afacan-Seref et al., 2018). This model could predict both behaviour and the temporal dynamics of neurophysiological signals reflecting decision formation. In particular, the model was able to capture a sudden, stimulus-evoked deflection in relative motor preparation initially towards the higher-value alternative, which, for low-value stimuli, was dynamically redirected toward the correct alternative. The model explained this by assuming that value biases are applied to sensory responses that are initially nonselective for color and become gradually more selective. In this way, the drift rate of the decision process, assumed to be driven by the difference in responses tuned to the two sensory alternatives, is initially dominated by value and later dominated by sensory information, as observed in the motor preparation dynamics.

Since the preceding model required assuming a two-phase sensory response that first detects (via nonselective activity) and then increasingly discriminates the sensory change being decided on, the question naturally arises, whether drift rate biases are peculiar to this situation, or are more generally invoked for any sensory feature when task demands require prioritisation. It has been shown that orientation-tuned neurons in the V1 region are immediately selective - that is, they respond quickly and vigorously to their preferred orientation and respond little if at all for the orthogonal orientation (Shapley, Hawken, & Xing, 2007). In this study, we therefore used an orientation discrimination task in order to assess the generality of drift rate biasing mechanisms when responding to sensory stimuli under time pressure.

Several lines of work have established that action selection dynamics at the motor level provide a key window onto the evolving decision process, because evidence accumulation is continuously fed to motor circuits (Selen, Shadlen, & Wolpert, 2012). For example, an fMRI study found a lateralized activation of the primary motor cortices since the very beginning of the evidence accumulation process (Gluth, Rieskamp, & Büchel, 2013). These authors also found that activity in the pre-supplementary motor area (pre-SMA) increased with time and was correlated with total accumulated evidence. This continuous involvement of the motor system during the decision-making process has been extended to value-biased decisions, where it has been shown that the Lateralized Readiness Potential (LRP), an event-related potential thought to reflect the relative degree of preparation to move the left versus right hand (Kornhuber & Deecke, 1965; Vaughan, Costa, & Ritter, 1968) reflects the ongoing process of evaluating the incoming sensory information from its very beginning (Noorbaloochi, Sharon, & McClelland, 2015).

Studies such as Gratton et al. (1988), Van Vugt et al. (2014) and Noorbaloochi et al. (2015) found evidence for static SPB signals reflected in the LRP component, which were strongly associated with response outcomes. In particular, Noorbalochi and colleagues showed that separate evidence related and reward related components could be clearly distinguished in the LRP signal. These features make this signal a great candidate for the analysis of our task.

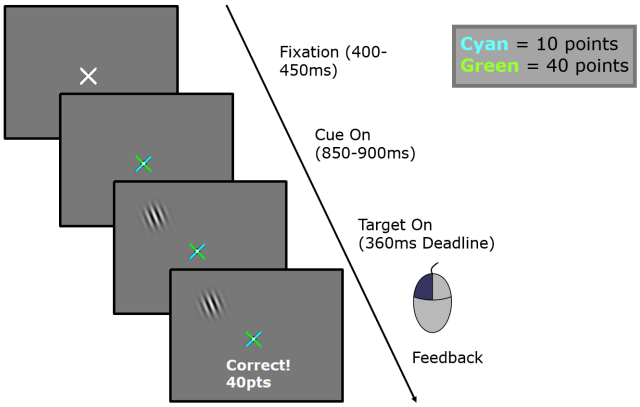


Figure 1: Orientation discrimination task.

In the present study we examined the static and/or dynamic biases at play in sensorimotor decisions under conditions of intense speed pressure. For this purpose, we recorded EEG activity during a value-biased orientation discrimination task under a strict deadline (Figure 1), where a correct response to one orientation was worth more (40 points) than to the other (10). We developed a bounded accumulation model informed by value-biasing signatures in the LRP and compared its fit to behaviour with that of existing models.

Method

A total of 25 participants took part in the study, but 3 were excluded from the analyses due to inadequate EEG signal quality. They were compensated with €32 for their participation and they could earn up to €12 depending on their performance. They all had normal or corrected-to-normal vision and gave informed consent to participate in the study which was approved by our local ethics committee.

In the task, after the initial fixation, a cue (two crossed, coloured lines) was presented at fixation to indicate which of the two alternatives was worth more points (40 vs 10) if it was to be presented and responded to correctly, though the orientation actually presented was equally likely. The trial's value

was randomised and each block contained an equal amount of High and Low value trials. 850-900ms following the cue, an oriented grating target appeared on the upper left or upper right of the screen (fixed within a block and counterbalanced across blocks). The appropriate amount of points was awarded if a left or right-hand response was made to a left-tilted or right-tilted grating, respectively, within a 360-ms deadline. The main EEG recording consisted of 8 blocks of 160 trials each.

Motor preparation was measured as the LRP (Gratton et al., 1988; de Jong, Wierda, Mulder, & Mulder, 1988; Eimer, 1998), at standard EEG sites C3 and C4. Four simplified versions of the bounded diffusion models were constructed to examine the two alternative value biasing mechanisms (SPB and DRB) and also to examine the mechanisms that produce fast errors (starting-point variability ['VS'] versus increasing evidence ['IE']) as was done in Afacan-Seref et al. (2018). The SPB-VS model was defined by the following equation:

$$\begin{aligned} x(t) &= x(t-1) + d \cdot dt + N(0, s\sqrt{dt}) \\ x(0) &\sim \mathcal{U}(\pm z_B, \pm z_B + s_z) \end{aligned} \tag{1}$$

Where d is the drift rate, dt is the discrete time increment (1 ms in simulations) and $N(0, s\sqrt{dt})$ refers to Gaussian noise with zero mean and variance $s^2 \cdot dt$. The SPB is z_B, with a positive sign for High Value and negative for Low Value trials. The starting point variability was determined by s_z. The DRB-VS model was defined by the following equation:

$$\begin{aligned} x(t) &= x(t-1) + (d \pm d_B)dt + N(0, s\sqrt{dt}) \\ x(0) &\sim \mathcal{U}(-s_z, +s_z) \end{aligned} \tag{2}$$

Where d_B is the symmetric bias in the drift rate (positive for High Value trials and negative for Low Value ones). The SPB-IE model was defined by:

$$\begin{aligned} x(t) &= x(t-1) + c \cdot t \cdot dt + N(0, s\sqrt{dt}) \\ x(0) &= \pm z_B \end{aligned} \tag{3}$$

Where c is the slope for the linearly increasing drift rate. The DRB-IE model was defined by:

$$\begin{aligned} x(t) &= x(t-1) + (c \cdot t \pm d_B)dt + N(0, s\sqrt{dt}) \\ x(0) &= 0 \end{aligned} \tag{4}$$

A fifth model was created inspired by the observed LRP dynamics (see Results), which included an early biased dynamic urgency signal. This model was defined by:

$$\begin{aligned} x(t) &= x(t-1) + u \cdot dt + e(t)\left(d \cdot dt + N(0, s\sqrt{dt}\right) \\ x(u_{oT}) &= 0 \\ e(t) &= \begin{cases} 0, & \text{if} \quad t < e_{oT} \\ 1, & \text{if} \quad t \geq e_{oT} \end{cases} \\ u &\sim N(\pm u_\mu, u_\sigma) \end{aligned} \tag{5}$$

Where u is the rate of increase of the urgency signal. The onset time of the urgency signal is defined by u_{oT}. The mean of this urgency signal has a positive sign for High Value conditions and a negative one for Low Value ones. The appearance of the evidence and start of the accumulation process is represented by the unit step function *e(t)* with onset at e_{oT}. For all models the decision variable x evolved with the stated dynamics until it crossed either an upper $(+1)$ or lower (-1) bound resulting in a correct or incorrect outcome, respectively, with the RT equated to the bound crossing time, such that any non-decision time is allowed for by the onset timing parameters u_{oT} and e_{oT}.

All models were fitted by Monte-Carlo simulation methods to individual participant choice and RT distributions with a bounded SIMPLEX routine (Nelder & Mead, 1965) implemented in the MATLAB function fminsearchbnd with a G^2 likelihood ratio statistic as the cost function, quantifying the divergence between the bins separated by the five quantiles $[.1, .3, .5, .7, .9]$ for correct and error trials in the simulated and real datasets.

Results

As expected, correct responses were more frequent on High Value than Low Value trials (90% versus 54%, $F(1,21) = 51.51, p < .001$; Figure 2). There was a significant interaction of value (high/low) x accuracy (correct/error) on RT (2 x 2 rmANOVA, $F(1,21) = 600.45, p < .001$), driven by the fact that correct trial RTs were significantly faster for High Value targets compared to Low Value targets ($t(21) = -76.773, p < .001$) and the opposite was observed for Errors $t(21) = 11.027, p < .001$). When plotting Accuracy over RT, a shift in the responses was observed for low value trials, from very fast, purely value-driven erroneous responses to slow sensory-driven correct ones (Figure 2). The fast value-driven responses are further emphasized by the perfect overlap in the leading tail of the distribution for Low value errors and High Value correct trials.

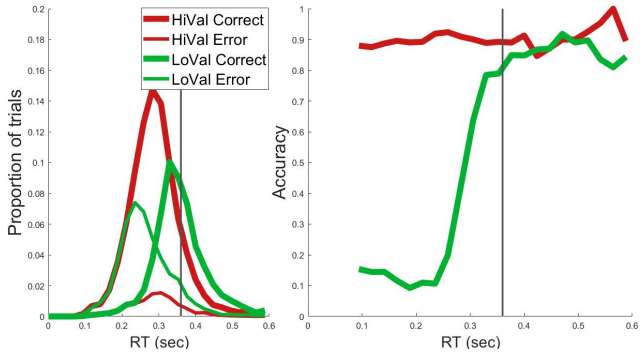


Figure 2: Response Time distribution and Conditional Accuracy Function averaged over participants.

A bias mechanism in the form of a SPB around target onset was observed in the LRP (Figure 3), across the dif-

ferent value conditions ($F(1,21) = 21.583, p < .001$) and it had an influence on choice outcome (for Low Value trials $F(1,21) = 6.88, p = 0.016$). Interestingly, this bias in starting level at target onset did not appear to be static, but continuously grew through the post-target delay period before the process accelerated due to bottom-up input. This reflects a dynamic urgency bias (value difference in the slopes of the target locked LRP from -100 to 100ms $t(21) = 4.6, p < .001$; Figure 3) and represents an empirical neural signature of DRB. We used this anticipatory urgency signal from the LRP to estimate an "urgency onset time" that we could compare to the urgency onset time estimated by our model (Table 1). A straight line was fitted to the target-locked LRP signal from -100ms to 100ms and extended backward in time until it reached zero, producing an estimate of the starting point of this dynamic bias: around 442ms before target onset. This empirical urgency onset time differed from the model's estimation (-442ms vs 72ms). Response-locked LRP plots showed a pre-response "threshold" level that did not significantly vary with value ($F(1,21) = .547, p = .468$) or location ($F(1,21) = .250, p = .622$). This was consistent with our model's assumption of constant bounds set above and below a one-dimensional decision variable.

Table 1: Estimated parameter values for the Urgency model, averaged across participants.

u_{oT}	u_{μ}	u_{σ}	d	s	e_{oT}
0.072	3.0342	4.205	9.973	2.839	0.27

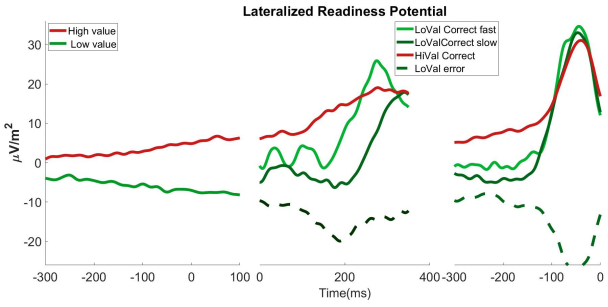


Figure 3: From left to right, LRP motor preparation dynamics time-locked to the target onset for High and Low value (left), separating Fast and Slow Low Value (center), and time-locked to the response (right), broken out by behavioural outcome. Upward deflections reflect preparation toward the correct response.

As in previous work (Afacan-Seref et al., 2018), a DRB model with increasing evidence showed a better fit to behaviour (lower Bayes Information Criterion [BIC], Figure 4) than SPB models, or models that included variability in their starting point rather than increasing evidence. However, our neurally-informed model, which instead incorporated an anticipatory biased urgency signal and time-invariant but later-onsetting evidence, provided the best fit overall (lowest BIC, Figure 4).

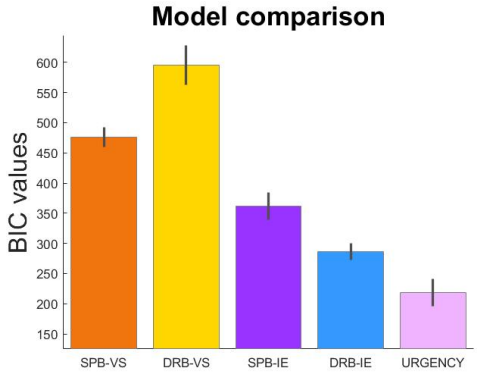


Figure 4: Mean BIC values quantifying goodness of fit. Error bars indicate S.E.M. after factoring out between-subjects variance.

Discussion

When faced with different environmental demands, choice-associated rewards need to be taken into account to make a decision that maximises expected gain. Within the stream of processes that lead to an action, at what point is this reward exerting its influence? What neural mechanism is responsible for it? Answering these questions and providing a neurally informed model that can capture behaviour in time-constrained situations is key to guiding leading theories about cognitive dysfunction in brain disorders such as ADHD, Autism, Depression, Addiction, Borderline Personality Disorder, Obsessive-Compulsive Disorder, and Parkinson's disease.

In this study, we examined behaviour and motor preparation dynamics during a two-choice rapid orientation discrimination task with asymmetric rewards (10 vs 40 points). Neurally informed mathematical modelling indicated that both static and dynamic biases are needed to explain behavioural data in fast decision scenarios and that the effects of both can be generated through a unitary mechanism, namely an anticipatory biased urgency signal.

Despite the prevalence of studies supporting static biases over dynamic ones for many years, recent studies are suggesting that dynamic biases might also play a role in the decision making process, especially when faced with time restrictions (Afacan-Seref et al., 2018). However, the literature already held examples of mixed results. On the one hand, for example, Ratcliff and McKoon (2008), used a biased motion discrimination task in which stimulus proportion for left or right responses was varied. The authors interpreted a shift in the leading edges of the RT distributions due to stimulus probability, as an indication of a SPB. This conclusion was also supported by their model, which showed that the differ-

ence in starting point accounted for most of the proportion effect. On the other hand, in one monkey single-cell recording study (Hanks et al., 2011) using a motion discrimination task, firing rates of decision variable encoding sensorimotor neurons built at a steeper rate toward the more probable option. They thus constructed a model that included a dynamically growing, evidence-independent bias component, which was able to account for human and monkey behaviour better than standard models. Although this bias signal would effectively implement a DRB, this interpretation was not universally accepted (Ratcliff, Smith, Brown, & McKoon, 2016). The model we constructed here is very similar but has the distinction that, as reflected in motor preparation dynamics, the biased urgency signal can, under time pressure, be well under way before the evidence begins to be processed. In doing so, we demonstrate that both starting point biases and drift rate biases are at play in the same scenario and can be generated by a single mechanism.

Accurately adjudicating between alternative cognitive models of the decision process can be difficult based on behaviour alone (O'Connell, Shadlen, Wong-Lin, & Kelly, 2018). In the present study, we used neural data to inform the structure and the fitting process of our models. A dynamic bias signal observed in the LRP, which started before evidence onset, effectively producing a SPB as well, indicated that an anticipatory urgency signal must have been at play. This is well expected in such a fast-paced task where the participant has very limited exposure to the stimulus before the deadline. Also, a unique threshold was observed in the motor preparation dynamics across conditions, inspiring the construction of a one-dimensional decision model.

It has been suggested that top-down expectations could influence the creation of representations or templates of the expected stimulus in the visual cortex, and these representations would later be compared to bottom-up stimulus information (Friston, 2005; Rao & Ballard, 1999; Mumford, 1992). Neural responses to redundant (expected) information in early sensory regions might be suppressed by higher-order regions (Mumford, 1992; Murray, Kersten, Olshausen, Schrater, & Woods, 2002; Rao & Ballard, 1999) or they could be rather increased by suppressing responses to stimuli that are inconsistent with the current expectations (Lee, Yang, Romero, & Mumford, 2002). Value biases could influence the creation of such representations as expectations do (Stanisor et al., 2013), by means of changing the drift rate during the decision making process. Alternatively, a modulation of the weighting or reference values used in the readout of these sensory representations could cause a change in the drift rate (Afacan-Seref et al., 2018). The present results can neither prove nor exclude the possibility that biases are also exerted at the sensory level or its readout. Further studies looking into an explicit modulation of the early visual cortical responses are needed in order to answer this question.

However, we found evidence for the third possibility mentioned above, a simpler mechanism that can produce drift rate biases that originate at the motor level (urgency). Up to now, motor preparation dynamics have been shown to reflect the continuous evaluation of incoming sensory information only from the beginning of the evidence accumulation process, assumed in standard models to onset after sensory encoding has been completed (Ratcliff & McKoon, 2008). However, our results suggest that decision-related motor preparation dynamics are in play well before the evidence is encoded, and that under time constraints, reward information is fed to the motor circuit to bias these anticipatory dynamics independent of the evidence presentation. This finding bears some similarity to Noorbaloochi et al. (2015) where separate evidence and reward related components were observed in the LRP, but our model is distinct in that rather than assuming two separate value-biased guess and sensory accumulation processes, reward information and sensory evidence jointly influence a single, dynamically evolving decision process. That there is a single, thresholded process is evidenced in the unique decision threshold that we observed in the LRP across value conditions (Figure 3).

In our study, unlike Noorbaloochi et al. (2015), the anticipatory urgency bias did correlate with response choice (see Results). This discrepancy could be caused by the different deadlines used in each study. Here, the deadline was very tight and to perform optimally you needed to start preparing before seeing the stimulus on the screen. In fact, our neural data suggest that this preparation started $\sim$ 442ms before stimulus onset and continuously grew over time, whereas in Noorbaloochi et al. (2015), the SPB was static, and their model estimated that the fast guess process onsets at $\sim$ 150ms after target onset. In fact, our own model's estimated parameter for the start of the urgency signal (0.072 s) did not coincide with our neural data. Even accounting for the fact that our onset estimate would be misestimated to be later by an amount equal to the motor non-decision time, the latter can be expected to be approximately 50-100 ms, leaving still a big discrepancy between the empirical and the estimated data. A possible next step could be to constrain the model, in order to match this urgency onset time and test whether it is still able to account for the behavioural dynamics. So far, the present model presents one unique mechanism that can qualitatively account for the increase in Starting Point Bias observed in the LRP and quantitatively capture the observed RT distributions.

Acknowledgments

Study funded by the Science Foundation Ireland (15/CDA/3591) and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 842143.

References

Afacan-Seref, K., Steinemann, N. A., Blangero, A., & Kelly, S. P. (2018). Dynamic Interplay of Value and Sensory Information in High-Speed Decision Making. *Current Biology*, *28*, 795–802. doi: 10.1016/j.cub.2018.01.071

de Jong, R., Wierda, M., Mulder, G., & Mulder, L. J. (1988). Use of Partial Stimulus Information in Response Processing. *Journal of Experimental Psychology: Human Perception and Performance*, *14*, 682–692.

Eimer, M. (1998). The lateralized readiness potential as an on-line measure of central response activation processes. *Behavior Research Methods, Instruments, and Computers*, *30*, 146–156.

Friston, K. (2005). A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *360*, 815–836.

Gluth, S., Rieskamp, J., & Büchel, C. (2013). Classic EEG motor potentials track the emergence of value-based decisions. *NeuroImage*, *79*, 394–403.

Gratton, G., Coles, M. G., Sirevaag, E. J., Eriksen, C. W., & Donchin, E. (1988). Pre- and Poststimulus Activation of Response Channels: A Psychophysiological Analysis. *Journal of Experimental Psychology: Human Perception and Performance*, *14*, 331–344.

Hanks, T. D., Kiani, R., & Shadlen, M. N. (2014, may). A neural mechanism of speed-accuracy tradeoff in macaque area LIP. *eLife*, *2014*.

Hanks, T. D., Mazurek, M. E., Kiani, R., Hopp, E., & Shadlen, M. N. (2011). Elapsed Decision Time Affects the Weighting of Prior Probability in a Perceptual Decision Task. *Journal of Neuroscience*, *31*, 6339–6352.

Hawkins, G. E., Forstmann, B. U., Wagenmakers, E. J., Ratcliff, R., & Brown, S. D. (2015). Revisiting the evidence for collapsing boundaries and urgency signals in perceptual decision-making. *Journal of Neuroscience*, *35*, 2476–2484.

Kornhuber, H. H., & Deecke, L. (1965). Hirnpotentialänderungen bei Willkürbewegungen und passiven Bewegungen des Menschen: Bereitschaftspotential und reafferente Potentiale. *Pflügers Archiv für die Gesamte Physiologie des Menschen und der Tiere*, *284*, 1–17.

Lee, T. S., Yang, C. F., Romero, R. D., & Mumford, D. (2002). Neural activity in early visual cortex reflects behavioral experience and higher-order perceptual saliency. *Nature Neuroscience*, *5*, 589–597.

Mumford, D. (1992). On the computational architecture of the neocortex - II The role of cortico-cortical loops. *Biological Cybernetics*, *66*, 241–251.

Murray, S. O., Kersten, D., Olshausen, B. A., Schrater, P., & Woods, D. L. (2002). Shape perception reduces activity in human primary visual cortex. *Proceedings of the National Academy of Sciences of the United States of America*, *99*, 15164–15169.

Nelder, J. A., & Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, *7*, 308–313.

Noorbaloochi, S., Sharon, D., & McClelland, J. L. (2015). Payoff information biases a fast guess process in perceptual decision making under deadline pressure: Evidence from behavior, evoked potentials, and quantitative model comparison. *Journal of Neuroscience*, *35*, 10989–11011.

O'Connell, R. G., Shadlen, M. N., Wong-Lin, K. F., & Kelly, S. P. (2018). Bridging Neural and Computational Viewpoints on Perceptual Decision-Making. *Trends in Neurosciences*, *41*, 838–852.

Rao, R. P., & Ballard, D. H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, *2*, 79–87.

Ratcliff, R., & McKoon, G. (2008). The Diffusion Decision Model: Theory and Data for Two-Choice Decision Tasks. *Neural Computation*, *20*, 873–922.

Ratcliff, R., Smith, P. L., Brown, S. D., & McKoon, G. (2016). Diffusion Decision Model: Current Issues and History. *Trends in Cognitive Sciences*, *20*, 260–281.

Selen, L. P., Shadlen, M. N., & Wolpert, D. M. (2012). Deliberation in the motor system: Reflex gains track evolving evidence leading to a decision. *Journal of Neuroscience*, *32*, 2276–2286.

Serences, J. T., & Saproo, S. (2010). Population response profiles in early visual cortex are biased in favor of more valuable stimuli. *Journal of Neurophysiology*, *104*, 76–87.

Shapley, R., Hawken, M., & Xing, D. (2007). The dynamics of visual responses in the primary visual cortex. *Progress in Brain Research*, *165*, 21–32.

Stanisor, L., Van Der Togt, C., Pennartz, C. M., & Roelfsema, P. R. (2013). A unified selection signal for attention and reward in primary visual cortex. *Proceedings of the National Academy of Sciences of the United States of America*, *110*, 9136–9141.

Van Vugt, M. K., Simen, P., Nystrom, L., Holmes, P., & Cohen, J. D. (2014). Lateralized readiness potentials reveal properties of a neural mechanism for implementing a decision threshold. *PLoS ONE*, *9*.

Vaughan, H. G., Costa, L. D., & Ritter, W. (1968). Topography of the human motor potential. *Electroencephalography and Clinical Neurophysiology*, *25*, 1–10.

Modeling Intrinsic Motivation in ACT-R: Focusing on the Relation Between Pattern Matching and Intellectual Curiosity

Kazuma Nagashima (nagashima.kazuma.16@shizuoka.ac.jp),
Junya Morita (j-morita@inf.shizuoka.ac.jp) ,
Yugo Takeuchi (takeuchi@inf.shizuoka.ac.jp)
Department of Informatics, Graduate School of Integrated Science and Technology, Shizuoka University,
3-5-1 Johoku, Naka-ku, Hamamatsu-shi, Shizuoka-ken, 432-8011 Japan

Abstract

To be keen learners, humans need both external and internal rewards. To date, many studies on environmental learning using intrinsic motivation for artificial agents have been conducted. In this study, we aim to build a method to express curiosity in new environments via the ACT-R (Adaptive Control of Thought-Rational) cognitive architecture. This model focuses on the "production compilation" and "utility" modules, which are generic functions of ACT-R, and it regards pattern matching with the environment as a source of intellectual curiosity. We simulated a path-planning task in a maze environment using the proposed model. The model with intellectual curiosity revealed that understanding of the environment was improved through the task of searching the environment. Furthermore, we implemented the model using a standard reinforcement learning agent and compared it with the ACT-R model.

Keywords: Cognitive modeling; intrinsic motivation; ACT-R

Introduction

Humans can learn in a wide range of environments to achieve their goals using rewards generated internally and externally. To simulate such keen learning through artificial agents, the concept of intrinsic motivation, which is driven by rewards, such as self-efficacy and curiosity, has been discussed by several researchers (Manoury, Sao, & Cédric, 2019; Schmidhuber, 2010; Singh, Barto, & Chentanez, 2005).

However, these researchers have not explained the connection of intrinsic motivation with other primitive cognitive functions based on the framework of reinforcement learning. In contrast, recent studies of cognitive modeling have increasingly relied on cognitive architectures, which integrate primitive functions commonly used in various tasks (see Kotseruba & Tsotsos, 2020, as a recent review). By sharing primitive processes between different tasks, the overall structure of human cognition is defined.

Following these trends of studies, the current study aims to present one of the possible cognitive mechanisms behind intrinsic motivation based on cognitive architecture. Among several cognitive architectures, we use ACT-R (Adaptive Control of Thought-Rational; Anderson 2007). This architecture has been widely used, and considerable research has been conducted on it. Furthermore, ACT-R has a module similar to that of reinforcement learning used in conventional autonomous agents. Thus, we consider it useful to model intrinsic motivation in ACT-R by connecting the basic cognitive functions that have already been validated by various psychological experiments.

Before presenting the model, we clarify our purpose by reviewing the previous studies relating to this topic. Following this, we propose a mechanism of intrinsic motivation, which especially focuses on pattern matching between the environment and internal knowledge, assuming a correlation to human intellectual curiosity. The proposed mechanism is implemented to run simulations of a specific task. Finally, we summarize the current status and indicate future directions of research.

Related Works

This section presents two directions of studies about environmental learning: studies based on reinforcement learning and ACT-R.

Intrinsic Motivation in Reinforcement Learning

To date, several researchers have studied artificial agents with intrinsic motivation (Manoury et al., 2019; Schmidhuber, 2010; Singh et al., 2005). These studies have modeled curiosity, which is one type of intrinsic motivation, and have investigated methods to make agents search the environment widely. Such studies have primarily used statistical learning frameworks, such as reinforcement learning. Usually, agents created from reinforcement learning determine their actions based on information received from the external environment. The environment generates rewards depending on the result of their actions, and they seek to maximize the rewards it over time. Regarding this traditional framework, Sutton and Barto (1998) pointed out that the boundaries between agents and the environment are not the same as the physical boundaries between the body and the environment. Following this claim, Singh et al. (2005) proposed intrinsically motivated reinforcement learning (IMRL). In contrast to conventional reinforcement learning, in which one receives a reward directly from the external environment, IMRL fluctuates depending on the state of the internal environment and models the curiosity for an unexpected response.

In recent years, this topic has remarkably progressed with a framework of deep reinforcement learning (Burda et al., 2018; Pathak, Agrawal, Efros, & Darrell, 2017). Burda et al. (2018) examined environmental learning based solely on intrinsic rewards. The screens of games, such as Atari and Unity maze tasks, were used as input (Mnih et al., 2015), and internal rewards were generated from novel experiences for agents. As a result, agents learned a wide range of environments and improved their game scores. The authors indicated that game environments are usually designed to stimulate the users' curiosity, and the game scores increase when they find

new information in the environment.

Environmental Search and Emotion in ACT-R

The studies presented in the previous section did not examine the association between humans and models but aimed to propose learning algorithms that realize optimal searches. In contrast, ACT-R is a cognitive architecture with modules corresponding to brain regions. For example, the declarative module retains experience and knowledge, and the goal module manages states in tasks. The production rules in ACT-R are selected based on the status of such modules, and they send commands to the modules as actions (e.g., search for knowledge that meets the conditions and update the current state of the task). These rules include variables that realize flexible correspondence (pattern matching) with module states.

Concerning environmental learning, Fu and Anderson (2006) used ACT-R to solve the repeated maze task by implementing knowledge concerning direction, such as up, down, left, and right. Their model used an ACT-R function called the "utility module", which is similar to Q-Learning, a model-free reinforcement learning algorithm to optimize a policy of determining action taken under a specific situation (Watkins, 1989). Using this module, the model received a positive reward when performing an action leading to achieving the current goal and a negative reward when performing an action that not leading to achieving the current goal. By doing this, the model learned to take optimal action by increasing the number of trials.

Other research performed path planning of the maze task using not only reinforcement learning but also learning of declarative knowledge, which was implemented in ACT-R. Reitter and Lebiere (2010) employed declarative knowledge representing the structure of mazes as topological maps and presented a backtracking algorithm searching the topological maps to find a goal. Their model did not include implementations of acquiring such topological maps but assumed that they were acquired through a general mechanism of instance-based learning that uses experience to solve the current situation (Gonzalez, Lerch, & Lebiere, 2003).

Although the official ACT-R theory has so far not directly included the topic of intrinsic motivation for environmental learning, many researchers have been working on models of emotion, which relate to intrinsic motivation. Dancy, Ritter, Berry, and Klein (2015) explained the influence of emotions by combining cognitive processes of ACT-R with physiological mechanisms. van Vugt and van der Velde (2018) built a cognitive model that describes depression by the proportion of memories with positive and negative emotions. Furthermore, Juvina, Larue, and Hough (2018) constructed a model of learning emotional memories using internally generated reward functions. Each of these studies developed novel modules or functions of ACT-R to approach emotional processes. In contrast, in this research, we aim to model intrinsic motivation using the existing built-in functions of ACT-R. In particular, the current study proposes a mechanism of reward fluctuation that naturally emerges from the learning process of ACT-R instead of directly defining reward functions as a formula.

Proposed Mechanism of Intrinsic Motivation

This section presents our proposed mechanism of intrinsic motivation. The mechanism is based on the idea of connecting intellectual curiosity with pattern matching. After describing this idea, we present a general framework of intrinsic motivation by combining the existing functions of ACT-R.

Intellectual Curiosity and Pattern Matching

Following Burda et al. (2018), we focus on *curiosity* as one of the causes of intrinsic motivation. As shown in previous studies, the agents' curiosity facilitates the exploration of the game environment, and the agents' game performance improves. In the book *Theory of Fun for Game Design*, game designer Koster (2004) said that good games stimulate users' curiosity. He also mentioned that the *fun* in the game is defined as *discovering patterns* leading to continuous learning. For example, in games where the optimal solution is found from several patterns, there is nothing to be obtained from the game after finding the optimal solution, and boredom occurs.

In the current study, we focused on the *pattern-matching* mechanism as a concept analogous to the discovery of patterns by humans. Pattern matching is a primitive-purpose mechanism. For a popular example, not limited to cognitive modeling, even text searching uses pattern matching, which is expressed as regular expressions. In ACT-R, as mentioned, pattern matching is used to match production rules and module states. Figure 1 explains pattern matching in ACT-R. In this example, *Variable 1* and *Variable 2*, which are included in the then clause of the ACT-R production rule are matched with the constants (i.e., numbers such as 1 and 2) of the declarative knowledge.

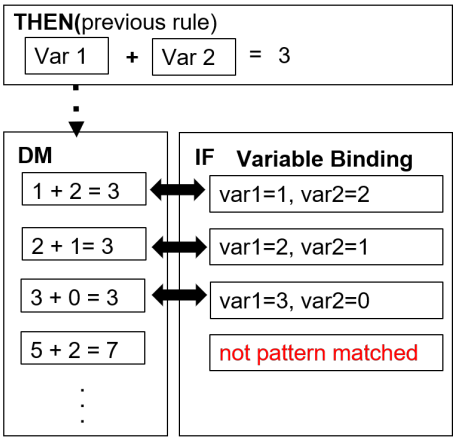


Figure 1: Example of ACT-R pattern matching. The model queries "declarative memory (DM)" with the "THEN" of the previous rule and illustrates the flow in which variables are bound by the "IF" of the next rule.

In this way, pattern matching discovers structures in the environment according to the patterns of variables embedded in the rules. Anderson (2007) claimed that a type of pattern matching dealing with relational structure is essential to achieving human-specific cognitive functions, such as cognitive flexibility, linguistic processing, metacognition, and analogical reasoning[1]. From this claim, we assume that the pattern matching of the cognitive model might lead to a model of human intellectual curiosity based on pattern discovery.

Decay of Intellectual Curiosity

To explain the role of intellectual curiosity in the primitive cognitive process, we need to consider how such a motivation decays during the process of task execution. We assume that such a decay process is the reverse of learning, namely boredom. To consider this in detail, the following subsections present a summary of the learning functions of ACT-R: the *utility* and *production compilation* modules.

Utility The ACT-R has two types of knowledge: declarative (chunks) and procedural (production rules). Each has learning mechanisms to acquire and to modulate the use of knowledge. Among these approaches, we focus on the modulation of procedural knowledge to determine whether to continue performing the task (motivated state) or to quit the task (bored state). As noted in the previous section, ACT-R has a utility module that controls a conflict resolution (selecting one of several rules that can fire (execute) in a specific situation) and updates the utilities through rewards (Fu & Anderson, 2006). Using this module, we solve a conflict between the task continuation rule and task stopping rule and assume that the number of rewards adjusting these utilities is influenced by the execution of the production compilation.

Production Compilation The production compilation module combines two successive production rules into one production rule (Taatgen & Lee, 2003). By repeatedly firing a series of rules for a certain task, the integration of rules occurs, and the number of rules that fire is reduced until the task is completed. Production rules that are the target of integration usually include variables in the conditional clauses (the IF parts). In ACT-R, the flexible nature of human thought is modeled by pattern matching of declarative knowledge with the pattern of variables described in a production rule. The integration of rules by production compilation skips such flexible pattern matching (i.e., avoiding retrievals of declarative memory). In other words, variables contained in the rules before integration are replaced by static values copied from declarative knowledge, and routine automatic task execution procedures are produced.

Figure 2 illustrates a trace of the ACT-R model that plans the path from the current position to the goal position in a maze environment, which is the task used in the simulation presented in the later section. The vertical axis indicates time, and each column indicates a module event. The trace on the left represents the initial state of the model using planning declarative knowledge to plan the path. The trace on the right is the process after the compilation when the model plans the path without retrieving declarative knowledge.

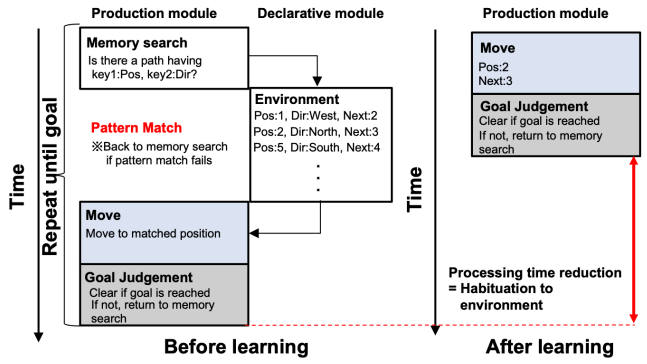


Figure 2: Example before and after learning using the *production compilation module*.

By applying this mechanism to the search of the maze environment, at first, the model often performs memory retrieval from the environmental map inside the declarative knowledge. As the task progresses, those memory retrievals becomes unnecessary. As a result, the model runs the tasks efficiently, and the frequency of pattern matching decreases due to the exhaustion of patterns in the environment.

Mechanism of the Task Continuation

Using the primitive functions presented so far, we propose our original mechanism of intellectual curiosity to determine whether to continue or to stop the task. Figure 3 illustrates a proposed mechanism of the continuation of a task in a general environment. At the start of each round, the model decides whether to continue or stop the task (conflict resolution between two rules). After it decides to continue the task, the model proceeds with the round by firing various rules (searching the map, etc.). When the model encounters a condition that ends the round, a new round is started, and the model decides whether to continue or stop the task again.

In the above process, the initial value of the utility of the *continue rule* is considered higher than that of the *stop rule*. At the beginning of the task, it can be assumed that humans intend to continue the task. The process of becoming bored from this initial state can be modeled by assigning a trigger of a negative reward to the rule that recognizes the end of each round. By triggering a negative reward at the end of the round, the utility of the *continue rule*, which have fired as a result of the previous conflict resolution, decreases, and the probability of firing the *stop rule* increases.

To prevent boredom and to consider the conditions for continuing environmental learning, a model of *intellectual curiosity*, namely *fun*, is required. If the model finds *fun* during the task, a positive reward is triggered, and the utility of

[1] Anderson (2007) made this claim at the introduction of the ACT-R function called *dynamic pattern matching*.

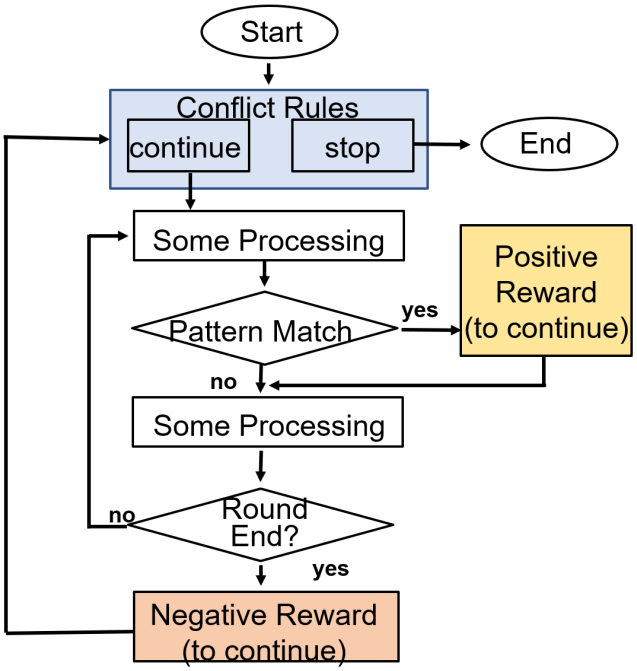


Figure 3: Flowchart of the task continuation model. Using "pattern matching" leads to a positive reward.

the *continue rule* maintains a high value. In this study, rules that trigger positive rewards are defined as rules that fire as a result of the successful retrieval of declarative knowledge in the task, such as remembering the map. The search for declarative knowledge requires pattern matching between the conditional clauses of the rule (the current situation) and the memory in declarative knowledge, and its success is consistent with Koster's definition of *fun* (finding patterns). However, this rule gradually becomes used to repeated execution; that is, the integration of rules occurs. After integration occurs, it becomes routine and cannot receive a reward. Then, the utility value of the *continue rule* decreases, and the *stop rule* fires. In short, long-term task continuation is achieved by keeping the model engaged in pattern matching between conditional clauses of production rules and declarative knowledge.

Implementation

The purpose of this study is to model intrinsic motivation by collecting primitive functions provided by ACT-R. For this purpose, it is necessary to implement the mechanism shown in the previous section on a specific task and observe its behavior. In this study, we select a path-planning maze task that has been used in many previous studies.

The Model of Maze Task

Our implementation of the ACT-R model for the maze search extends the memory-based strategy described in Reitter and Lebiere (2010) to include the mechanisms of task continuation (Figure 3).

Environment Figure 4 illustrates the maze in the present research. Reitter and Lebiere (2010) represented the maze environment as a topological map, consisting of a collection of cell IDs and links between cell IDs. In ACT-R, such a topological map is represented by a collection of chunks, and the model searches the environment, retrieving these chunks in the declarative memory.

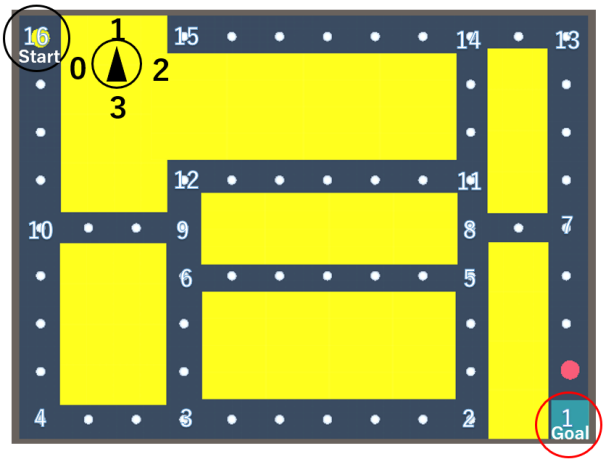


Figure 4: Maze environment

Searching Behaviours During the task, the model stores the current cell ID in the goal buffer. The model is initially located in #16 in Figure 4, and the model changes this state to #1 by retrieving a chunk stored in the declarative module. The chunk associated with the current position is requested, and the goal buffer is updated if the model retrieves such a chunk. This procedure is repeated until the model reaches the goal (#1).

Each time the goal is reached, the model labels all the chunks used in the current round as the correct path and stores these in the declarative module. From the next rounds, the model runs the task efficiently using these labeled chunks, following the method of the instance-based learning theory (Gonzalez et al., 2003).

If the model fails to retrieve the correct chunks, the model plans the path from the current position to the goal position using a heuristic search, namely a stochastic depth-first search (DFS). To realize backtracking used in a DFS, we implemented a stack structure using the imaginal module of ACT-R. Figure 5 depicts the stack function using chunks generated by this module. The push function in the stack is realized by generating a chunk that stores the name of the past chunk in the *ARG*1 slot. In addition, the pop function in the stack is realized by returning the *ARG*1 slot value to the past slot value. These generated chunks are stored in the declarative knowledge and can be retrieved later to realize the pop function. We implemented all these processes only through ACT-R production rules without defining any external functions written in other programming languages, such as LISP.

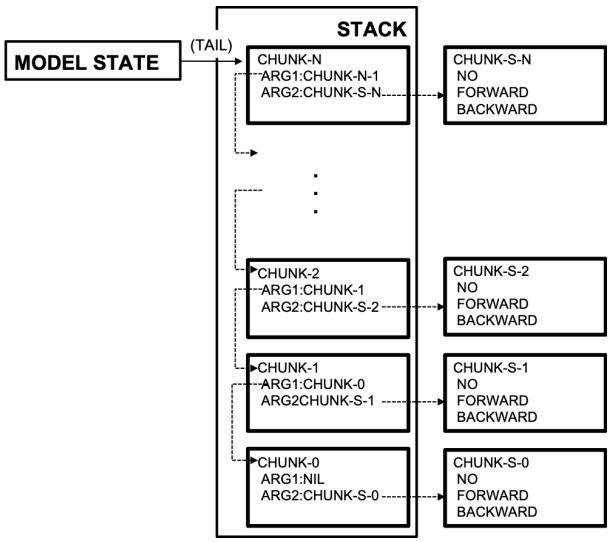


Figure 5: Stack structure of chunks implemented using the "imaginal module" of the ACT-R. During the search, the state of the model is dynamically stored (pushed) in the declarative module. The stored chunks are retrieved (popped) by following the dependencies noted in the slots of the chunks (dashed arrows).

Intrinsic Motivation in the Task In this simple maze task, we tried to observe how *fun* and *boredom* occur. In this model, *fun*, which is attached to pattern matching, is defined as remembering the correct path from the current situation to the goal. Specifically, the success of the DFS is defined as the attenuation of the motivation for continuing the task (positive rewards). In contrast, regardless of the success or failure of the goal search, a rule that fires at the end of the round is used as a trigger for a negative reward. The utility value of the *continue rule* decreases, as the negative trigger associated with the end of the round continues to occur without the rule expressing fun firing during the round. When the utility value of the *continue rule* falls below the utility value of the stop rule, the *stop rule* fires, and the task is terminated.

Simulation

Settings To confirm the behavior of our model of intrinsic motivation, we performed a simulation where the initial utility value of the *continue rule* was set to 10, and the initial utility value of the *stop rule* was set to 5. We also assigned the triggers of the negative reward ($r = 0$) to rules that recognized the end of the round (reaching goal #1 or recognizing that the time limit of each round has passed) and assigned the triggers of the positive reward to rules that included pattern matching. In this research, we select the *path finding rule* rule (DFS success) as the trigger of the positive reward, varying the value from 1 to 20 as the simulation conditions. For each condition of the positive reward value, the model runs the task 1000 times. In addition, we set the time limit of each round from 100 to 300 s. When the time limit was reached, the model resolved the conflict between the *continue* and *stop rules*.

The model also has rules that stochastically determine the directions to proceed (up, down, left, and right). The initial values of these utilities were also set to 10. Following Anderson et al. (2004), noise parameters were set as follows: ans (activation noise level) = 0.4 and, egs (production noise level) = 0.5.

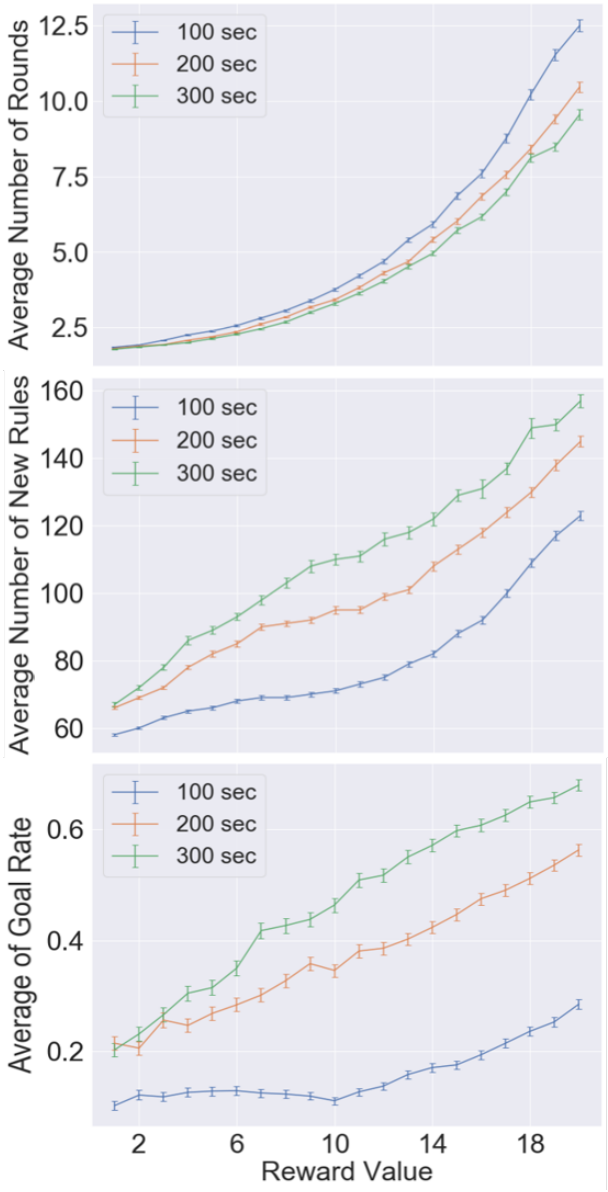


Figure 6: Results of the ACT-R model. Top: number of continued rounds; Middle: new rules generated by the production compilation; Bottom: goal rates of the model. The error bars in each graph represent the standard error.

Results Figure 6 displays the results of the simulation. From this figure, we observe that the reward generated by the pattern matching increased the number of continued rounds,

number of rules generated by the compilation module, and goal achievement rate. These results indicate that the implemented intrinsic motivation, which makes the model have a longer task continuation, leads to acquiring richer knowledge, resulting in better performance.

Comparison with Conventional Method

To further clarify the behavior of the proposed ACT-R model of intrinsic motivation, we compared it with a reinforcement learning agent that searched the maze environment in Figure 4. In this simulation, we used the algorithm of IMRL (Singh et al., 2005). At each time point, the reinforcement learning model is located on one of the numbered positions in the map, and it moves to up, left, down, or right. The selection of the direction is controlled with IMRL with ε-greedy ($\varepsilon = 0.2$, $\gamma = 0.9$, $\alpha = 0.2$):

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\acute{r_i} + \acute{r_e} + \gamma max Q(\acute{s},\acute{a}) - Q(s,a)] \quad (1)$$

Equation 1 indicates the updating of the Q value of the present model, where r_e represents a reward from the external environment, and r_i represents a reward from the internal environment. The model receives $r_e = -1$ when it fails to remember the path, whereas it receives $r_e = 0$ when it successfully remembers the path. The model also obtains $r_e = 10$ when it reaches the goal. Contrary to r_e, r_i is determined with Equation 2.

$$r_i = -\tau(1-p)\log(1-p) \quad (2)$$

where p represents the transition probability with the Q value. From this equation, the internal reward r_i is determined as the entropy of the probability of the complementary event with respect to p. In addition, τ is the coefficient for the reward value used in the simulation. A larger τ indicates a greater intrinsic motivation. In this study, we manipulated this value from 0.34 to 0.74. For each condition of the τ value, the model runs the task 1000 times. We manipulated the number of steps of movement in each round (100, 125, and 150 steps). When the step limit was reached, the model chose to quit the task or to continue the task by comparing the summation of the obtained internal reward (r_i) with the given threshold ($th = 5$).

Figure 7 indicates the result of the simulation of the reinforcement learning model. Similar to the ACT-R model, the internal reward increases the number of rounds. However, in contrast to the ACT-R model, it slightly decreases the goal rate. That is, the reinforcement learning model with high intrinsic motivation does not learn to achieve the goal but learns to explore the environment. This behavior might be changed by modulating the balance between r_i and r_e in Equation 1 or by designing the maze environment carefully to stimulate curiosity, as suggested by Burda et al. (2018). However, our ACT-R model could learn the environment without such careful parameter modulations or environmental design. Therefore, from this simulation, we can claim the advantage of our model in representing intrinsic motivation.

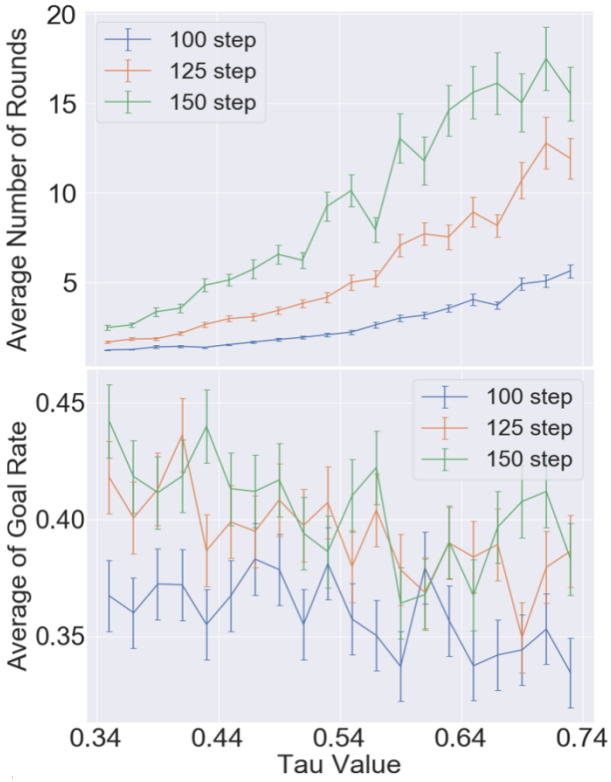


Figure 7: Results of the reinforcement learning model. Top: number of continued rounds; Bottom: goal rates. The error in each graph is the standard error.

Conclusion

The purpose of this study was to construct a model of intrinsic motivation by accumulating primitive cognitive processes provided by ACT-R. To achieve this goal, we assumed that the mechanism of pattern matching represents the source of intellectual curiosity, namely fun. Thus, with the success of pattern matching, the model maintains high intrinsic motivation for task continuation. In contrast, by skipping the pattern matching with compilation mechanisms, the model 'tires' of the task and eventually stops.

From the simulation results presented in Figures 6 and 7, we consider that our model has an advantage in learning new environments. The model uses both the utility module and instance-based learning (memorizing the correct path to the goal; Gonzalez et al., 2007). Such a combination of several learning algorithms might help balance the intrinsic and extrinsic rewards in the current maze task.

However, the result in the previous section does not indicate that the conventional reinforcement learning cannot achieve the same learning as ACT-R. The model presented by Singh et al. (2005) included the mechanism called option, which summarizes low-level actions into abstract-level units (Sutton, Precup, & Singh, 1999) and indicated the process of moving up using abstract *option* as the model learned the environment. This mechanism has a commonality with

the compilation module in ACT-R. Schmidhuber (2010) also pointed out that such a compression mechanism is the same as the prediction mechanism, which leads to the emotional process of fun and boredom. We need to further explore the relationship between such models of reinforcement learning and the proposed model.

In addition to the efficacy of environmental learning, the expression of the internal reward of our model has an advantage compared to previous studies. In our model, we did not explicitly divide the internal and external rewards in the equation, but the effect of intrinsic motivation is represented in the existing mechanism of ACT-R. We consider that this approach has an advantage because it is based on the theory of human cognition, it is related to the existing learning research, and it saves unnecessary factors in the theory.

In future studies, we need to compare the model of intrinsic motivation with human data. As a model of human cognition, behavior presented by conventional reinforcement learning might not be wrong. During search tasks, people often forget the goals and decrease in performance. Such irrational behavior might also relate to *computational psychiatry* (Huys, Maia, & Frank, 2016).

We also need to model the optimal level of motivation (Yerkes & Dodson, 1908). In this study, the model statistically determines the initial utility value of the *continue rule* to focus on the decay process of intellectual curiosity. The process up to the optimal level, obtaining intrinsic motivation for the target environmental learning, is not modeled. Therefore, by constructing a model representing such a process, we can explore more detailed conditions of task continuation, especially those before the model reaches optimal levels.

Acknowledgment

This research was supported by JSPS KAKENHI Grant Numbers 17H05859, 20H05560, 20H04996.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe*. Oxford Press.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., & Efros, A. A. (2018). Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*.

Dancy, C. L., Ritter, F. E., Berry, K. A., & Klein, L. C. (2015). Using a cognitive architecture with a physiological substrate to represent effects of a psychological stressor on cognition. *Computational and Mathematical Organization Theory*, *21*(1), 90–114.

Fu, W.-T., & Anderson, J. R. (2006, 6). From recurrent choice to skill learning: A reinforcement-learning model. *Journal of Experimental Psychology. General*, *135*, 184-206.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*(4), 591–635.

Huys, Q. J., Maia, T. V., & Frank, M. J. (2016). Computational psychiatry as a bridge from neuroscience to clinical applications. *Nature Neuroscience*, *19*(3), 404.

Juvina, I., Larue, O., & Hough, A. (2018). Modeling valuation and core affect in a cognitive architecture: The impact of valence and arousal on memory and decision-making. *Cognitive Systems Research*, *48*, 4 - 24.

Koster, R. (2004). *Theory of fun for game design*. Paraglyph Pr.

Kotseruba, I., & Tsotsos, J. K. (2020). A review of 40 years of cognitive architecture research: Focus on perception, attention, learning and applications. *AI Review*, *53*, 17-94.

Manoury, A., Sao, M. N., & Cédric, B. (2019). Hierarchical affordance discovery using intrinsic motivation. In *In Proceedings of the 7th International Conference on Human-Agent Interaction (HAI 19)* (pp. 186–193).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 16–17).

Reitter, D., & Lebiere, C. (2010). A cognitive model of spatial path-planning. *Computational and Mathematical Organization Theory*, *16*(3), 220–245.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, *2*(3), 230–247.

Singh, S., Barto, A. G., & Chentanez, N. (2005). Intrinsically motivated reinforcement learning. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems 17* (pp. 1281–1288). MIT Press.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors*, *45*(1), 61–76.

van Vugt, M. K., & van der Velde, M. (2018). How does rumination impact cognition? A first mechanistic model. *Topics in Cognitive Science*, *10*(1), 175–191.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. King's College, Cambridge.

Yerkes, R. M., & Dodson, J. D. (1908). The relation of strength of stimulus to rapidity of habit-formation. *Journal of Comparative Neurology and Psychology*, *18*(5), 459–482.

Effects of Decision Complexity in Goal-seeking Gridworlds: A Comparison of Instance-Based Learning and Reinforcement Learning Agents

Thuy Ngoc Nguyen (ngocnt@cmu.edu) and Cleotilde Gonzalez (coty@cmu.edu)
Dynamic Decision Making Laboratory
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh PA 15213 USA

Abstract

Decisions under uncertainty are often made by weighing the expected costs and benefits of the available options. The costs-benefits tradeoffs may make decisions easy or difficult, particularly given uncertainty of these costs and rewards. In this research, we evaluate how a cognitive model based on Instance-Based Learning Theory (IBLT) and two well-known reinforcement learning (RL) algorithms learn to make better choices in a goal-seeking gridworld task under uncertainty and on increasing degrees of decision complexity. We also use a random agent as a base level comparison. Our results suggest that IBL and RL models are comparable in their accuracy levels on simple settings, although the RL models are more efficient than the IBL model. However, as decision complexity increases, the IBL model is not only more accurate but also more efficient than the RL models. Our results suggest that the IBL model is able to pursue highly rewarding targets even when the costs increase; while the RL models seem to get "distracted" by lower costs, reaching lower reward targets.

Keywords: decision complexity; instance-based learning theory; reinforcement learning; goal-seeking task.

Introduction

Goal-seeking in gridworld navigation, has long been a classical task for developing Artificial Intelligent (AI) agents. Generally, the agent must navigate an environment (e.g., gridworld) with uncertainty about the surroundings to achieve a goal (i.e., consuming the highest rewarding object) given a number of obstacles and within a time limit. This type of task underlies a broad range of applications such as search and rescue or pickup and delivery missions.

Researchers have commonly addressed this type of task using Reinforcement Learning (RL) models, a computational method of learning from interaction (Sutton, Barto, et al., 1998; Gershman & Daw, 2017). A major challenge for research in AI is to develop systems that can replicate human behavior; and although there is much evidence of RL's ability to account for human behavior in some dynamic decision tasks (Gureckis & Love, 2009; Simon & Daw, 2011), the concern has been raised that the advance in RL paradigms is mostly centered on solving computational problems efficiently, rather than replicating or explaining in detail how humans learn (Botvinick et al., 2019).

Cognitive modeling on the other hand, is aimed at understanding and interpreting human behavior by representing the cognitive steps by which a task is performed. In particular, Instance-based Learning Theory (IBLT) was developed to provide a cognitively-plausible account for how humans make decisions from experience and under uncertainty, through interactions with dynamic environments (Gonzalez, Lerch, & Lebiere, 2003). IBLT has shown accurate representation of human choice and broad applicability in a wide number of decision making domains, from economic decision making to highly applied situations, including complex allocation of resources and cybersecurity, e.g. (Hertwig, 2015; Gonzalez, 2013; Gonzalez et al., 2003).

Nevertheless, in goal-seeking gridworld navigation tasks, cognitive models of decision making, and IBL models in particular, have been less common. Fu and Anderson (2006) proposed a RL mechanism within the ACT-R architecture, to account for repeated choice and skill learning. The study used a maze learning task, and showed that the model can fit human data fairly well to account for complex learning in this task. Also, Reitter and Lebiere (2010) proposed an ACT-R cognitive model, to address the aspects of human path-planning problems, which are relatively similar to navigation. Finally, in a prognostic foraging task, Chelian and colleagues showed that both IBL models and RL approaches can imitate human decision making well (Chelian, Paik, Pirolli, Lebiere, & Bhattacharyya, 2015). Despite all of these advances, it remains unclear how RL and IBL models compare with respect to representing human decisions under uncertainty.

To that end, the primary goal of this work is to examine how RL and IBL agents learn in a goal-seeking gridworld task under different degrees of decision complexity. Decisions under uncertainty are often made by weighing the expected costs and benefits of the available options. Some decisions are easy (e.g., choosing between an option of low cost and high expected reward and one with high cost and low reward), while others are complex (e.g., choosing between low cost low reward, and high cost and high reward options). These decisions' complexity increases given uncertainty in the costs and rewards. Thus, we first leverage IBLT, to develop an IBL model of an agent that is able to accomplish the goal-seeking task in a gridworld environment under different levels of decision complexity. Using simulation experiments, we explore the impact of decision complexity on the performance of different types of agents, RL and IBL, including a Random that serves as a baseline comparison for the models.

Instance-Based Learning Theory

IBLT is a theory of decisions from experience, developed to explain human learning from interaction with dynamic decision environments (Gonzalez et al., 2003). IBLT provides an algorithm and a set of cognitive mechanisms that can be used to implement computational models of decision learning processes. The algorithm involves the recognition and retrieval of past experiences (i.e., instances) according to their similarity to a current decision opportunity. Instances retrieved are

used to calculate the expected utility of a potential decision in such situation. Potential decision alternatives a are evaluated sequentially, and a process of choice provides a stopping point for evaluating potential alternatives and making a choice. The choice alternative with the highest expected utility among a set of alternatives is selected. Finally, a feedback process updates the expected utility of past instances with the observed actual outcome of choices executed. Such updated instances are then reused in future decisions.

An "instance" in IBLT is a memory unit, that results from the potential alternatives evaluated. These are memory representations consisting of three elements: a situation (S) (set of attributes that give a context to the decision, or state s); a decision (D) (the action taken corresponding to an alternative in state s, or action a); and a utility (U) (expected utility u or experienced outcome x of the action taken in a state). The essential sub-symbolic mechanisms of IBLT have been discussed in multiple past publications (e.g. (Gonzalez et al., 2003; Gonzalez & Dutt, 2011; Gonzalez, Ben-Asher, Martin, & Dutt, 2015; Hertwig, 2015)), but we include these mechanisms here for completeness.

Each instance i in memory has a value of *Activation*, which represents how readily available that information is in memory (Anderson & Lebiere, 2014). The instance could be perfectly or partially matched to the attributes of a decision opportunity at the current point of time, which is determined by the partial matching mechanism (Anderson & Lebiere, 2014). But here we consider a simplified version of the Activation equation which only captures how recently and frequently the considered instances are activated:

$$A_i = \ln\left(\sum_{t' \in \{1..t-1\}} (t-t')^{-d}\right) + \sigma \ln \frac{1-\gamma_i}{\gamma_i}, \qquad (1)$$

where d and σ are respectively the decay and noise parameters; t' refers to the previous timestamp in which the outcome of instance i was observed resulting from choosing an action a at state s. The rightmost term represents the Gaussian noise for capturing individual variation in activation, and γ_i is a random number drawn from a uniform distribution $U(0,1)$.

Activation of an instance i is used to determine the probability of retrieval of such instance from memory. The probability of an instance i is a function of its activation A_i relative to the activation of all other instances corresponding to executing action a at state s:

$$p_i = \frac{e^{A_i/\tau}}{\sum_l e^{A_i/\tau}}, \qquad (2)$$

where τ is the Boltzmann constant (i.e., the "temperature") in the Boltzmann distribution (Kittel, 2004).

For simplicity, we defined τ as a function of the same σ parameter used in the activation equation $\tau = \sigma\sqrt{2}$. The parameter τ gives some variability to the probability of retrieving instances from memory.

The expected utility of taking action a in state s is calculated based on a mechanism called *Blending* (Lebiere, 1999) as specified in IBLT (Gonzalez et al., 2003), using the past experienced outcomes stored in each instance x. Here we employ the blended value that was defined and used for binary choice tasks in Lejarraga, Dutt, and Gonzalez (2012); Gonzalez and Dutt (2011):

$$V(a,s) = \sum_{i=1}^{n} p_i x_i. \qquad (3)$$

Essentially, according to (Gonzalez & Dutt, 2011), Blending (Equation 3) is the sum of all the past experienced outcomes weighted by their probability of retrieval, where x_i is the outcome stored in an instance i associated with taking action a at state s; p_i is the probability of retrieving the instance i from memory (Equation 2); and n is the number of instances stored in memory for taking action a up to the last trial.

The choice rule is to select the action a that corresponds to the maximum blended value.

Goal-seeking Task in Gridworld Environment

A gridworld environment is made up of a 11×11 grid maze as illustrated in Figure 1. Each gridworld contains randomly-located obstacles (black bars). The number of obstacles varies from one to five and their size ranges from one to six 1×1 cells. There are four targets of different values, which are represented as four colored objects (blue, green, orange, and purple) of size of 1×1 in the grid and set at random locations in a way that does not overlap with the obstacles.

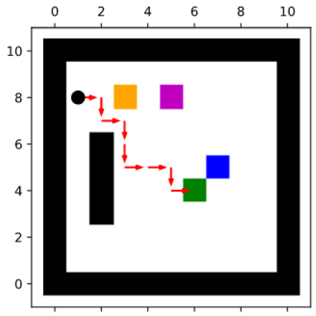


Figure 1: Illustration of the goal-seeking task in the gridworld environment. The agent's preferred goal is the "green" object.

The primary task is a goal-seeking problem in the gridworld environment, where an agent (black dot), starting in a random location (i.e., (x,y)), moves through the 11×11 grid to search for the most valuable goal among the four objects, while avoiding obstacles. The agent is tasked with consuming the object that has the highest reward (i.e. "green" in Figure 1) within a 31 step limit. Starting in its initial position, the agent makes sequential decisions about which actions to take (i.e., up, down, left, right). An episode ends when the agent decides to "consume" *any* of the four objects, or by reaching the 31 step limit without a consumption.

A sequence of moves from the initial location to the end location forms a *trajectory* (dotted red line) which is produced

by the sequence of decisions that the agent adopts. Each agent performs the task over 500 episodes for learning in the same gridworld.

Technically, each agent $\mathcal{A}_k$ is driven by a fixed reward, $r_{k,j} \in (0,1)$, for consuming an object o_j where $j = 1,\ldots,4$. Hence, the vector $r_k = (r_{k,1},\ldots,r_{k,4})$ has four components (one for each of the four objects), and it was drawn from a Dirichlet distribution ($\sum_{j=1,\ldots,4} r_{k,j} = 1$ and $r_{k,j} > 0$) with concentration parameter $\alpha = 0.01$, which signified that the agent $\mathcal{A}_k$ was favourably attracted to one of the four objects. In other words, if the agent successfully consumes the most preferred object, it will receive the highest reward while consuming any of the other 3 objects (i.e. the distractors) results in receiving much smaller rewards. Besides, the agent is penalized 0.01 for each step, which is a movement cost, and 0.05 for walking into a wall.

Agents in the Gridworld

IBL Agent

In the gridworld task, an *instance* is defined by a triplet (s,a,x), where x is the outcome or expected utility resulting from taking action a (i.e., up, down, left right) in state s (i.e., the state is the location of the agent, defined by the x-y coordinates) in a grid (Nguyen & Gonzalez, 2020). When making a prediction about which action a the agent $\mathcal{A}_k$ will take at state s, the IBL agent selects the action with the highest expected utility using the *blended* value (Equation 3).

Importantly, the agent only gets a positive outcome when consuming an object after a sequence of decisions. Thus, the IBL agent must learn to update the expected utility from the outcome received after consuming an object, so that different instances created by the trajectory are reinforced accordingly. The delayed feedback mechanism proposed in IBLT (Gonzalez et al., 2003) is underdeveloped, and most of the tasks that IBLT has been applied to, include immediate feedback. Thus, a mechanism to deal with delayed feedback is required in the gridworld task. Unlike prior work that focused on how humans learn from delayed feedback (Walsh & Anderson, 2011; Kelly & West, 2013), we simply use the final outcome and distribute it equally to all actions taken in a trajectory. That is, considering the trajectory $\mathcal{T}_k = \{(s_t,a_t)\}_{t=0}^{T}$ if the $\mathcal{A}_k$ gets the outcome x' at the end of the episode ($t = T$) then the expected utility of executing $\{(s_t,a_t)\}_{t=0}^{T-1}$ is all updated to x'. We leave the alignment to human judgements of delayed feedback for future research.

RL Agents

We compare the performance of the IBL agent against two RL agents called Q-learning and SARSA which are well-known temporal difference techniques in RL (Sutton et al., 1998). The basic difference between these two RL algorithms is in the way of updating a value of current state-action pair. In SARSA (*on-policy* method), the update takes into account the value of the actual action taken at one state ahead of the current state whereas in Q-learning (*off-policy* method), it simply considers the highest possible action that can be taken at the current state.

Q-learning Agent. A Q-learning agent was implemented with a tabular form of Q-learning algorithm (Sutton et al., 1998). In general, the goal of the RL agent $\mathcal{A}_k$ is to estimate the optimal state-action values referred to as Q-values, where $Q(s,a)$ returns the expected future reward of action a at state s. Initially, all the Q-values are set to zero and then are iteratively updated. Given enough iterations, the agent can learn the optimal Q-values denoted by $Q^*(s,a)$, and for each state s the agent selects the action having the highest Q-value, $\pi_k^*(s) = \text{argmax}_a\, Q^*(s,a)$.

SARSA Agent. A SARSA agent was designed based on the SARSA algorithm. The name SARSA comes from the fact that the updates depend on a quintuple of events (s,a,r,s',a'), where s and a are the current state and action of the agent, r is the observed reward for choosing the action a, and s' and a' are the new state-action pair. Essentially, SARSA, in contrast to Q-learning, learns the value of each state–action pair (i.e. the Q-value) by looking ahead to the next action to see what the agent will perform at the next step and then update the Q-value of its current state-action pair accordingly.

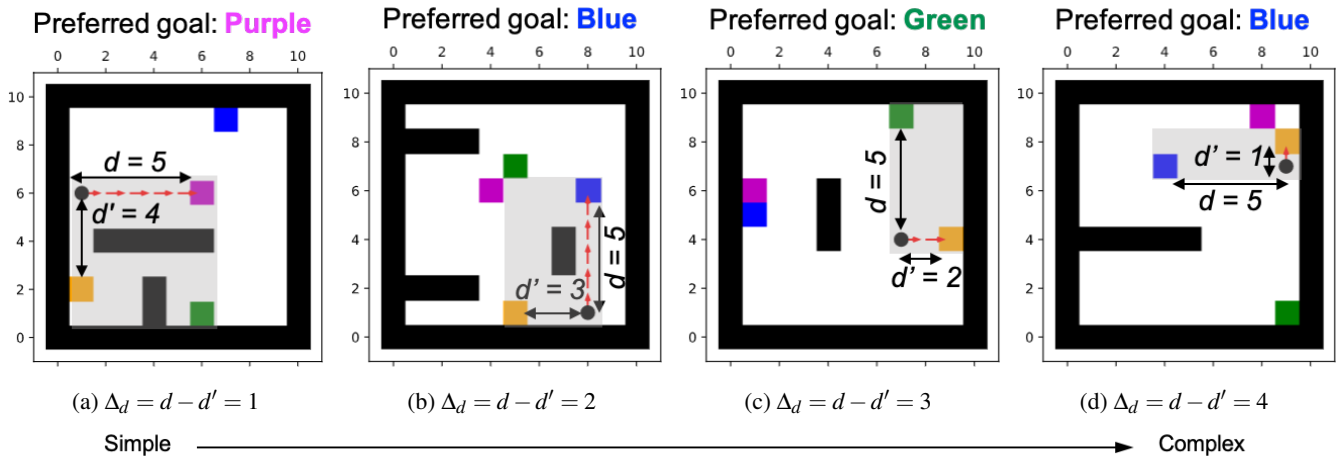
Random Agent

A random agent $\mathcal{A}_k$ selects an action a in state s based on the probability $\pi_k(a|s)$. Precisely, the policy of $\mathcal{A}_k$ is drawn from a Dirichlet distribution $\pi_k \sim Dir(\alpha)$ with concentration parameter α, so that $\sum_{a\in A} \pi_k(a|s) = 1$ and $\pi_k(a|s) > 0$. If α is close to 0 then the policy of an agent is characterized to be near deterministic. Conversely, the action of the agent is far more stochastic if α is much greater than 0.

Experiments

To investigate how different agents perform under different levels of the decision complexity, we designed experimental manipulations in which we control cost-benefit tradeoffs of choices made in a gridworld task.

Inspired by general decision processes and animal behavior, we designed levels of complexity. In animal foraging the complexity involves a tradeoff between the quality of food and the effort of obtaining it, and this tradeoff also applies to human decision processes (Mehlhorn et al., 2015). A gridworld can be more complex when its arrangement of goals and obstacles creates a high conflict between benefits (i.e., the object's reward) and associated costs (i.e., the distance, or number of steps needed to consume that object). For instance, a setting in which an agent must decide whether to consume a close (e.g., one step distant from the current agent's location) but low-reward object or to search for a far-away but higher reward object is more challenging than a decision between a close and high-reward object and a far and low-reward object. We refer to low-reward objects as "distractors" and to the highest reward object as the "preferred object". Agents



(a) $\Delta_d = d - d' = 1$ (b) $\Delta_d = d - d' = 2$ (c) $\Delta_d = d - d' = 3$ (d) $\Delta_d = d - d' = 4$

Figure 2: Illustration of the designed gridworlds with the level of complexity increasing from left to right.

generally prefer high value objects but they need to explore the environment to learn the value of the four objects, since this is only known after they consume an object.

In our experiment, complexity is characterized by the difference between the distance from an agent to the preferred object (d) and the distance from the agent to the closest distractor (d'), i.e., $\Delta_d = d - d'$. Intuitively, the larger the value of Δ_d, the more complex the decision is, given the temptation to consume a closer distractor than to consume the highest value distant goal. Simply put, the high value of Δ_d signifies the high conflict between consuming the preferred object with the longer distance d or the distractor with the shorter distance d'.

The experiment design is illustrated in Figure 2. It is worth noting that we only examine the cases when $\Delta_d > 0$ as when $d > d'$. Take Figure 2a as an example of how the setup works. Here the distractor is the "orange" object while the highest value goal is "purple". The distance from the agent's location to its goal is $d = 5$ and to the distractor object is $d' = 4$, and hence $\Delta_d = (d - d') = 1$. This is a simple environment since the cost to reach the highest value goal and the distractor is nearly equal (and thus, preferring the highest value goal over the distractor is a simple choice). In contrast, as exemplified in Figure 2d, with $\Delta_d = 4$, the choice is more complex, since preferring the highest value goal ("blue" object) is more costly than consuming the distractor ("yellow" object) and as a result, the agent may be attracted to the closer object (even if the reward is lower).

Model Parameters

The IBL agent's parameters are $\sigma = 0.25$ and $d = 0.5$, default parameters that come from the ACT-R architecture (Anderson & Lebiere, 2014). For the Random agent, we consider $\pi_k \sim Dir(\alpha = 3)$. Regarding the parameters of Q and SARSA, we set the discount factor $\gamma = 0.99$ and the learning rate $\alpha = 0.1$.

Independent Variables

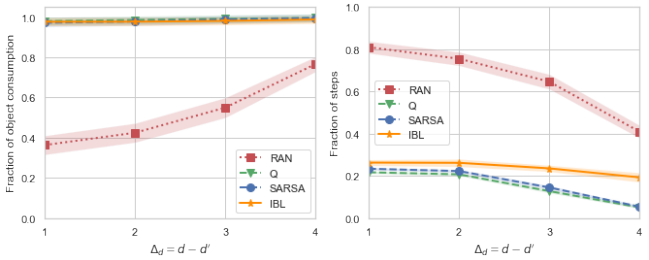
For simplicity, in this experiment we deal only with the trade-off between one preferred goal and one distractor, where the distance between an agent and its preferred goal is fixed to $d = 5$. Hence, to manipulate decision complexity, we only vary the distance from the agent to the distractor ($d' = 1 \ldots 4$). We examined four levels of decision complexity ($\Delta_d = 1..4$) and four types of agents (IBL, Q, SARSA, and Random). For each of the four levels of complexity, we ran 100 agents of each type, that is, for each value of Δ_d, 100 different gridworlds were generated. In each gridworld, the agents had 500 learning episodes.

Evaluation Metrics

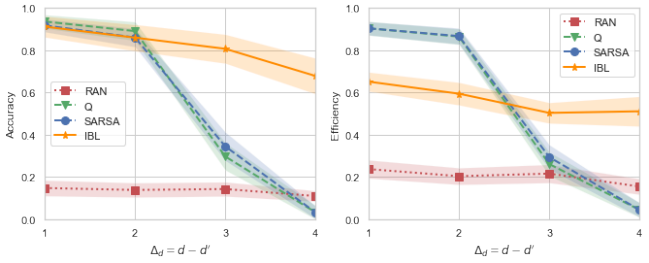
For each model we calculated the following measures: (1) *Fraction of object consumption*: the proportion of episodes (out of 500) in which the agent reaches one of the four objects (i.e., rather than wandering around and reaching the limit of steps without consuming any object); (2) *Fraction of steps*: the average ratio (across 500 episodes) between the number of steps for consuming any of the four objects and the maximum number of steps; (3) *Accuracy*: the proportion of episodes (out of 500) wherein the agent accomplishes the task (i.e. successfully consumes the highest value goal); and (4) *Efficiency*: the ratio between the reward from consuming an object and the movement cost (i.e., the multiplication of the penalty for each step and the number of steps taken) across the 500 episodes. The efficient values were normalized to the range in $[0, 1]$ using min-max normalization, i.e. $(value - min)/(max - min)$.

Results

We have analyzed the performance of the four types of agents, namely IBL, Q, SARSA, and Random, with respect to complexity ($\Delta_d = 1 \ldots 4$).



(a) Fraction of obj. consumption (b) Fraction of steps



(c) Accuracy (d) Efficiency

Figure 3: Performance of the agents in the task when varying the degree of complexity $\Delta_d = 1 \ldots 4$ (X-axis).

Fraction of Object Consumption

Figure 3a shows that the Fraction of object consumption for IBL and the two RL agents (Q and SARSA) reaching *any* of the objects is approximately equal to 1 regardless of complexity. Unsurprisingly, the Random agent performed considerably worse than the other agents, but their capability to consume an object in less than 31 steps increased with complexity. This can be explained by the environment design that the more challenging the environment is, the closer the agent is to a distractor. Hence, it is evidently easier for a Random agent to bump into an object when Δ_d increases.

Fraction of Steps

Figure 3b shows the Fraction of steps, suggesting that the IBL agent took about the same steps to get an object regardless of the level of complexity, while the Q and SARSA agents took slightly less steps with larger complexity. We also observe that the Random agent required the most steps to find an object, but the fraction decreases as complexity increases. Again, the most likely explanation is that the agents tend to consume the closer distractors.

Accuracy

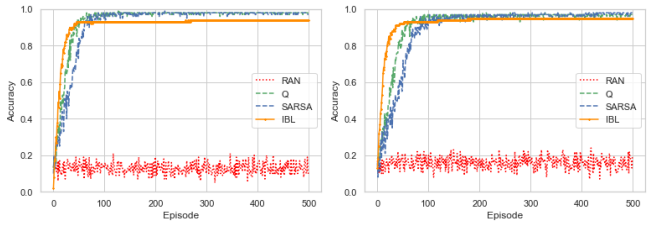
Figure 3c demonstrates that Accuracy of the RL agents and the IBL agent are comparable in simple environments ($\Delta_d = 1$ and 2). However, when complexity increases ($\Delta_d = 3$ and 4), IBL exhibits only a light drop in accuracy, while the Q and SARSA agents dropped accuracy significantly, reaching close to random accuracy with the highest complexity. More concretely, the IBL agents with an approximate 70% overall success rate by far surpassed the Q and SARSA agents whose the fraction of successful episodes was less than 5% over all 500 episodes. With respect to the Random agent, its curve is flat and nearly constant at about 0.18 over the values of Δ_d, signifying that its performance is independent of complexity due to its random characteristic.

Efficiency

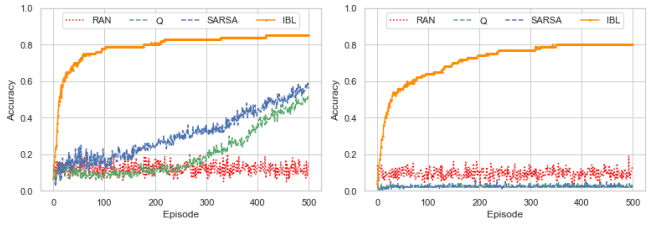
Figure 3d reveals that the Q and SARSA agents are the most efficient agents in the simple decisions ($\Delta_d \leq 2$), followed by IBL and then by Random. The higher value of the ratio between the benefit (i.e. the consumption reward) and the movement cost (i.e. the penalty for each step that the agent takes $\times$ the number of steps) indicates that the RL agents are able to obtain an object having the highest reward within a limited number of steps, when the decision is simple. Conversely, in complex decisions ($\Delta_d > 2$) the results show that the IBL agent is the most efficient, followed by the RL agents and the Random agent. The Efficiency together with the Accuracy results suggest that the RL agents are "distracted" by the closer objects and end up consuming the closer objects rather than affording the costs of searching for the highest value object. As a result, they got a significantly small amount of reward.

Learning Curves of Accuracy

To start to explain the observations above regarding the accuracy of the models, we analyzed the average Accuracy for each type of agent over the course of 500 episodes. This analysis would help observe how the accuracy developed within each level of complexity. Figure 4 demonstrates that the IBL agent learned slightly faster than the RL agents even in lower levels of complexity. The learning speed of the IBL models decrease with increased complexity, but the difference between IBL and the RL agents is larger in the complex settings ($\Delta_d > 2$). The Random agent does not learn.



(a) $\Delta_d = d - d' = 1$ (b) $\Delta_d = d - d' = 2$



(c) $\Delta_d = d - d' = 3$ (d) $\Delta_d = d - d' = 4$

Figure 4: Learning curves of the agents over 500 episodes for each level of complexity Δ_d.

Specifically, in the most complex decision environment ($\Delta_d = 4$), the average Accuracy achieved by the IBL agent

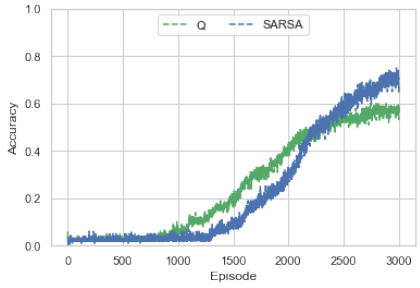


Figure 5: Learning curves of the Q and SARSA agents over 3000 episodes when $\Delta_d = 4$.

was over 0.6 just after 100 episodes, while the average Accuracy of the RL agents was nearly zero. To investigate this further, we ran the RL models for 3000 episodes under the highest level of complexity ($\Delta_d = 4$). The results shown in Figure 5 demonstrate that the Q and SARSA agents have a low start but learn to be more accurate in the highest complexity levels, after extended practice. We speculate that the one-step update of state-action values in the RL algorithms may prevent them from learning faster and determining the value of the various objects within 500 attempts. In contrast, the IBL model uses all the past instances in the blending mechanism (but these instances are decayed to different degrees as in Equation 1). This aggregation of more experiences may help to evaluate the decision tradeoffs more accurately, resulting in faster and more successful weigh of the costs and benefits in the decisions.

Conclusions

We investigated the performance of an IBL agent, two RL agents (Q-learning and SARSA), and a Random agent, while performing a navigation task under uncertainty and under increasing decision complexity. The decision complexity is formalized as the tradeoff between the objects' rewards and the associated movement costs. To select the object to consume in the presence of uncertainty, the agents must evaluate the expected reward of the object and the steps needed to reach it (costs), from experiential learning.

Experimental results revealed that the Accuracy and Efficiency of the two RL agents were not robust to increased levels of decision complexity, while the IBL cognitive model was more resilient to higher levels of complexity. The explanation is that the one-step update of state-action values in the RL agents results in these agents getting "distracted" by near objects, which are consumed even when they are of lower value. Thus, as the difficulty of the decisions increases the Accuracy and Efficiency of the RL agents decrease. The IBL agent is less efficient than the RL agents under low levels of complexity but under higher complexity levels it learns to consume the higher value objects even when it takes more steps to reach those objects.

Acknowledgments

This research is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), award number: FP00002636. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

References

Anderson, J. R., & Lebiere, C. J. (2014). *The atomic components of thought*. Psychology Press.

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., & Hassabis, D. (2019). Reinforcement learning, fast and slow. *Trends in cognitive sciences*.

Chelian, S. E., Paik, J., Pirolli, P., Lebiere, C., & Bhattacharyya, R. (2015). Reinforcement learning and instance-based learning approaches to modeling human decision making in a prognostic foraging task. In *2015 joint ieee international conference on development and learning and epigenetic robotics* (pp. 116–122).

Fu, W.-T., & Anderson, J. R. (2006). From recurrent choice to skill learning: A reinforcement-learning model. *Journal of experimental psychology: General*, *135*(2), 184.

Gershman, S. J., & Daw, N. D. (2017). Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annual review of psychology*, *68*, 101–128.

Gonzalez, C. (2013). The boundaries of instance-based learning theory for explaining decisions from experience. In *Progress in brain research* (Vol. 202, pp. 73–98). Elsevier.

Gonzalez, C., Ben-Asher, N., Martin, J. M., & Dutt, V. (2015). A cognitive model of dynamic cooperation with varied interdependency information. *Cognitive science*, *39*(3), 457–495.

Gonzalez, C., & Dutt, V. (2011). Instance-based learning: Integrating decisions from experience in sampling and repeated choice paradigms. *Psychological Review*, *118*(4), 523–51.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*(4), 591–635.

Gureckis, T. M., & Love, B. C. (2009). Short-term gains, long-term pains: How cues about state aid learning in dynamic environments. *Cognition*, *113*(3), 293–313.

Hertwig, R. (2015). Decisions from experience. *The Wiley Blackwell handbook of judgment and decision making*, *1*, 240–267.

Kelly, M. A., & West, R. L. (2013). Decision making in a dynamically structured holographic memory model: Learning from delayed feedback. In *Proceedings of the international conference on cognitive modelling*.

Kittel, C. (2004). *Elementary statistical physics*. Courier Corporation.

Lebiere, C. (1999). Blending: An act-r mechanism for aggregate retrievals. In *Proceedings of the sixth annual act-r workshop*.

Lejarraga, T., Dutt, V., & Gonzalez, C. (2012). Instance-based learning: A general model of repeated binary choice. *Journal of Behavioral Decision Making*, *25*(2), 143–153.

Mehlhorn, K., Newell, B. R., Todd, P. M., Lee, M. D., Morgan, K., Braithwaite, V. A., ... Gonzalez, C. (2015). Unpacking the exploration–exploitation tradeoff: A synthesis of human and animal literatures. *Decision*, *2*(3), 191.

Nguyen, T. N., & Gonzalez, C. (2020). Cognitive machine theory of mind. In *Proceedings of the 42nd annual meeting of the cognitive science society (cogsci 2020)*.

Reitter, D., & Lebiere, C. (2010). A cognitive model of spatial path-planning. *Computational and Mathematical Organization Theory*, *16*(3), 220–245.

Simon, D. A., & Daw, N. D. (2011). Environmental statistics and the trade-off between model-based and td learning in humans. In *Advances in neural information processing systems* (pp. 127–135).

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning* (Vol. 2) (No. 4). MIT press Cambridge.

Walsh, M. M., & Anderson, J. R. (2011). Learning from delayed feedback: neural responses in temporal credit assignment. *Cognitive, Affective, & Behavioral Neuroscience*, *11*(2), 131–143.

The Cognitive Modeling of Errors During the Japanese Phonological Awareness Formation Process

Jumpei Nishikawa (nishikawa.jumpei.16@shizuoka.ac.jp)[1]
Junya Morita (j-morita@inf.shizuoka.ac.jp)[1]

[1]Faculty of Informatics, Shizuoka University, 3-5-1, Johoku, Naka-ku, Hamamatsu, Shizuoka 432-8011,Japan

Abstract

Both nature and nurture contribute to language development. In the case of phoneme segmentation, children have the natural ability to recognize a continuous sound in various units, but as they grow, they only selectively learn to recognize it as part of a series in the unit that is used in their mother tongue. This developmental process is supported by an ability called phonological awareness that allows children to become intentionally aware of units of phonology. It is known that erroneous pronunciation appears during the phonological awareness formation process. In this research, we aim to examine the factors that induce and reduce such errors. To do so, we modeled phonological awareness using the cognitive architecture ACT-R and performed simulations that manipulated ACT-R parameters that correspond to both nature and nurture factors. As a result, it was confirmed that errors due to a lack of phonological awareness can be modeled with the innate memory retrieval mechanism. We also observed that such errors were reduced when learning factors were added to the model. However, we could not simulate this learning process. In the future, we will study the interaction task that enables learning to reduce phonological errors and contribute to the acquisition of phonological awareness.

Keywords: cognitive modeling; phonological awareness; Japanese;

Introduction

Language development is influenced by both innate and experiential factors. Newborn infants have an innate foundation for acquiring a variety of languages. Through experiences such as observing and imitating the behavior of familiar adults (Baron-Cohen, 1997; Tomasello, 1999), they gradually form a specific mother language structure, based on a general, innate language foundation. A prominent example of such convergence can be found in phoneme segmentation, which is one of the linguistic components. Infants initially have opportunities to divide a sound into various types of segments, like syllables or morae, though, as they grow up and master their language, they find it difficult to recognize phoneme segments that do not belong to their mother tongue.

In the fields of developmental psychology and speech-language pathology, this developmental process is partly attributed to an ability called phonological awareness, which enables one to intentionally pay attention to phonological aspects, such as phonemes and rhythms, of oral languages (Stahl & Murray, 1994). This ability is usually developed during the preschool period and relates to the acquisition of reading and writing skills. Researchers have detected several phonological errors that seem to occur due to a lack of phonological awareness during the language development process (Kubozono, 1989). In addition, there are reported cases in which infants with atypical developmental traits, such as those on the autism spectrum, who experienced an overall delay in acquiring phonological units, improved these problems with support for enhancing their phonological awareness (Mugitani et al., 2019).

Although psychologists have revealed the age at which phonological awareness is formed and its role in language development, it has not been clarified that the internal processes, cognitive functions, and mechanisms behind the occurrence and suppression of errors relating this ability.

Based on this background, the current study aims to examine error occurrence and suppression factors that are related to the phonological awareness formation process. We focus on two language development factors, namely nature and nurture, and construct a model using the cognitive architecture ACT-R (Anderson, 2007). In addition, we examine the process and mechanism of the phonological awareness formation process by simulating Japanese language development.

The structure of this paper is as follows. The first section introduces the research related to this study. The model and the model-based simulation will then be shown. Finally, a summary of the current status and future issues is presented.

Related Research

In this section, we first introduce previous studies on language development and the formation of phonological awareness. Next, we introduce cognitive modeling and cognitive architecture as the method for understanding and explaining human cognitive processes. We also review research on language learning using cognitive models.

Studies on Phonological Awareness

In the field of phonology, researchers have developed systems for classifying the sounds that humans perceive and utter as mental representations of sounds. The most famous system is the distinctive feature (Chomsky & Halle, 1968), which is the basic unit that distinguishes phonemes; it is typically defined as a binomial variable that takes the value of $+$ or $-$ by classifying the tongue and throat movements associated with vocalization. Table 1 provides the distinctive features for some phonemes.

Table 1: Distinctive feature examples

	a	i	u	e	o	k	g	s	z	t	d	n
consonantal	−	−	−	−	−	+	+	+	+	+	+	+
syllabic	+	+	+	+	+	−	−	−	−	−	−	−
sonorant	+	+	+	+	+	−	−	−	−	−	−	+
hign	−	+	+	−	−	+	+	−	−	−	−	−
back	+	−	+	−	+	+	+	−	−	−	−	−
low	+	−	−	−	−	−	−	−	−	−	−	−
anterior						−	−	+	+	+	+	+
coronal						−	−	+	+	+	+	+
voice						−	+	−	+	−	+	+
continuant						−	−	+	+	−	−	−
nasal						−	−	−	−	−	−	+
strident						−	−	+	+	−	−	−
delayed release						−	−	−	−	−	−	−
round	−	−	+	−	+	−	−	−	−	−	−	−

Around the world, there are varieties of systems that combine these innate phonetic elements as units. According to Trubetzkoy (1969), such systems fall into two groups, namely "mora languages" and "syllable languages," based on the smallest prosodic unit that is used in that language. Among the mora languages, Japanese defines the mora as a unit of duration (Bloch, 1950), and each mora is associated with a single *kana* (Japanese character), which consists of five vowels (*a, i, u, e, o*) and 59 combinations of consonants and vowels (*ka, ki, ku, ke, ko, sa, si, etc.*), and other special morae representing the duration or gemination of sounds. In other words, Japanese phonic elements have a clear connection with written symbols. We think that this characteristic offers an advantage for modeling phonological awareness in a symbolic cognitive architecture. Therefore, in this study, we focus on Japanese morae and try to reveal factors that relate to the acquisition process of this ability.

Several Japanese researchers have investigated the formation of phonological awareness. Hara (2001) presented a study where participants engaged in several phonological manipulation tasks. For example, in the mora deletion task, participants were required to respond with a mora sequence that involved deleting a single mora from an orally presented word (e.g., *i-ko* is the correct answer for a task that deletes *ta* from the orally presented word *ta-i-ko*, which means "a drum"). Similarly, in the mora reversal task, participants were required to respond with a reverse-ordered mora sequence after an oral word prompt (e.g., *ka-i-su* for *su-i-ka*, which means "a watermelon"). In her study, the performance of such tasks was increased according to participants' written and reading language skills.

In a clinical setting, Japanese speech-language-hearing therapists (ST) reported cases where children confused specific types of morae. For example, it was reported that young children around 2 to 3 years old tend to confuse morae that include the consonants /r/ and /d/ or /s/ and /sj/, though Japanese adults can clearly distinguish these (Kobayashi, 2018). Moreover, there have been reports that some children with developmental disorders have difficulty distinguishing a mora that consists of only a vowel as well as other morae that include that vowel (e.g., confusions between *a* and *ka*, *sa*, or *ta*) (Ishida & Ishizaka, 2016). Typically developed children also exhibited similar erroneous utterances in which they omitted consonants (e.g., pronouncing *a-p-pa* instead of *ra-p-pa*; trumpet) (Nakamura, Kojima, & Fujiwara, 2015). This type of confusion suggests the existence of a developmental process that migrates from the innate phonological system (Chomsky & Halle, 1968) to the language-specific mora system.

Furthermore, several researchers have used the popular Japanese word game *"shiritori"* as a task for examining a development stage of phonological awareness. In this game, players take turns providing a noun whose first character (mora) is the same as the last character of the previously given noun. For example, after a player answers *"ri-n-go"* (meaning: apple), the other player continues with *"go-ri-ra"* (meaning: gorilla). The game is over when a player provides a word that ends with /N/, since no Japanese word can begin with this character (e.g., *"ri-n-go"* $\rightarrow$ *"go-ri-ra"* $\rightarrow$ *"ra-p-pa"* (trumpet) $\rightarrow$ *"pa-n"* (Bread) $\rightarrow$ game over). To avoid looping, the game also ends if a player repeats a word that was already provided as an answer in the game (e.g., *"ma-su-ku"* (mask) $\rightarrow$ *"ku-ru-ma"* (car) $\rightarrow$ *"ma-su-ku"* $\rightarrow \cdots$).

Takahashi (1997) examined the conditions for being able to play *shiritori* through a cross-sectional developmental psychological experiment involving children with typical development. This research indicated that playing *shiritori* requires the ability to divide sounds into morae and the mental lexicon indexed by phonemes, and that the acquisition of kana characters is effective for indexing vocabulary by morae. These results suggest that playing *shiritori* requires phonological awareness, paying attention to a phoneme in the mother language (mora) in a sound, and that such ability is enhanced by presenting visual aids (kana character) that correspond to the sound. Furthermore, it has been shown that even if a child does not have the phonological awareness that is necessary to play *shiritori*, s/he can participate with help from adults, who can provide hints.

Kubozono (2000) also conducted an experiment that examined the *shiritori* process in a 4-year-old Japanese child in order to present the phonological awareness formation process. In his experiment, the participant was sometimes confused about the unit of mora. For example, she gave *"mo-n-shi-ro-cho-u"* (cabbage butterfly) as an answer for *"do-ra-e-mo-n"* (doraemon). She also noted *"yo-u-gu-ru-to"* (yogurt) after *"ta-i-yo-u"* (sun) was presented. Based on these two examples, the participant seems to have recognized the consonant-vowel-vowel sequence (*"mo-n"* and *"yo-u"*) as a single unit, although the Japanese mora system divides this into two morae, namely "consonant-vowel" and "vowel." These reports suggest that the language-specific phonetic system is experientially acquired after childhood, and the misconfiguration of the system can be observed culturally while playing popular word games.

Based on the above-mentioned previous findings, the current research focuses on the ability to extract the mora at the end of a word from the continuous sounds that correspond

to words and the ability to search for words by initial mora, especially in phonological awareness. In addition, we apply *shiritori* as the task set based on Takahashi's and Kubozono's work. In addition, we refer to Chomsky and Halle (1968)'s definition and use a distinctive feature to express and characterize the mora as a symbol.

Cognitive Modeling

One of the methods for understanding and explaining the mechanisms and processes that are related to human cognition is through the cognitive modeling approach in the field of cognitive science. In this approach, a model that approximates a person's task execution (a cognitive model) is built as a computer program, and a simulation is done using the model. By observing the cognitive model's behavior and internal states during the simulation, we can infer a person's internal states and cognitive processes during task execution.

Cognitive architectures have been developed as a basis for cognitive modeling. Using a cognitive architecture, it is possible to construct a model that isolates the factors related to task achievement on a common basis. From among the various cognitive architectures that have been developed, we selected ACT-R (Anderson, 2007) for use in this study. Since ACT-R is based on psychological experiments on thinking and memory, it enables us to comprehensively grasp various phenomena related to human cognition. Furthermore, ACT-R is a production system with multiple modules. There are various parameters that define the module's behavior, thus making it easier to model the individual. There are also modules that handle interaction with the external environment, thus making it possible to predict reaction times and associate them with experimental data.

There have been many studies on language acquisition using ACT-R. For instance, models related to the acquisition of irregular verbs in English learning (Taatgen & Anderson, 2002) and models of infants' noun learning (Van Rij, Van Rijn, & Hendriks, 2010) have been constructed. In addition, brain dysfunction has also been modeled by ACT-R, and some studies have explained the errors in sentence comprehension in aphasia using ACT-R parameters (Mätzig, Vasishth, Engelmann, Caplan, & Burchert, 2018).

In this study, we aim to model the language development process by mapping phonological awareness from the memory retrieval mechanism of the ACT-R declarative module. In particular, we focus on parameters that express knowledge similarity and the effect of learning, assuming that changes in these parameters through simulation correspond to innate, experiential language development factors.

Model

In this section, we describe the model of errors in the Japanese phonological awareness formation process. The model executes *shiritori* to exhibit reported errors and demonstrate the factors that suppress such errors.

An overview of the model is shown in Figure 1. This model includes two agents (dashed line area) who keep a game of *shiritori* going by taking turns providing words. Boxes in the agent area corresponds to each module of ACT-R. In the following, we show how the *shiritori* process is realized through the ACT-R module structure.

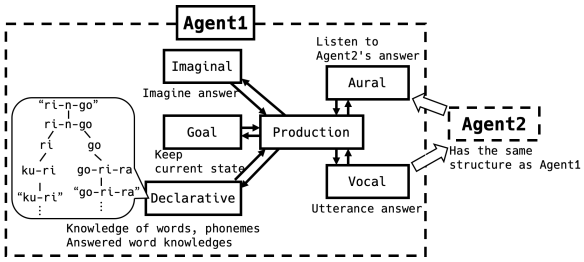


Figure 1: An overview of the model

Module Structure

Declarative Module The declarative module of ACT-R contains knowledge that is required for task execution in the form of chunks. The model in this study retains three types of chunks that relate to word (vocabulary) and phonological knowledge, and the association between words and phonological knowledge (Table 2). The model also has chunks that store the words that have already been provided in the current *shiritori* trial. These chunks do not exist in the declarative module at the beginning of the trial; rather, they are generated and stored as the trial progresses using imaginal module.

Table 2: The model's declarative memory

(a) Word knowledge

word	sound
ringo	"ringo"
gorira	"gorira"
kuri	"kuri"
...	...

(b) Phonological knowledge

mora	sound
/ri/	"ri"
/go/	"go"
/ku/	"ku"
...	...

(c) The word–mora relationship

word	mora	position
ringo	/ri/	head
gorira	/go/	tail
gorira	/go/	head
...	...	...

Production Module The production module selects and applies rules, and operates the module, while using information and states that are held by other modules. In the model of this research, when word information is received as a partner's answer, the novel word is retrieved and provided as an answer according to the rules of *shiritori* that were presented in the previous section.

Figure 2 summarizes this process. Using the word chunk (chunk type b in Table 2) acquired by the aural module, the model retrieves a chunk that connects the word and the ending mora (chunk type c in Table 2). Phonological knowledge

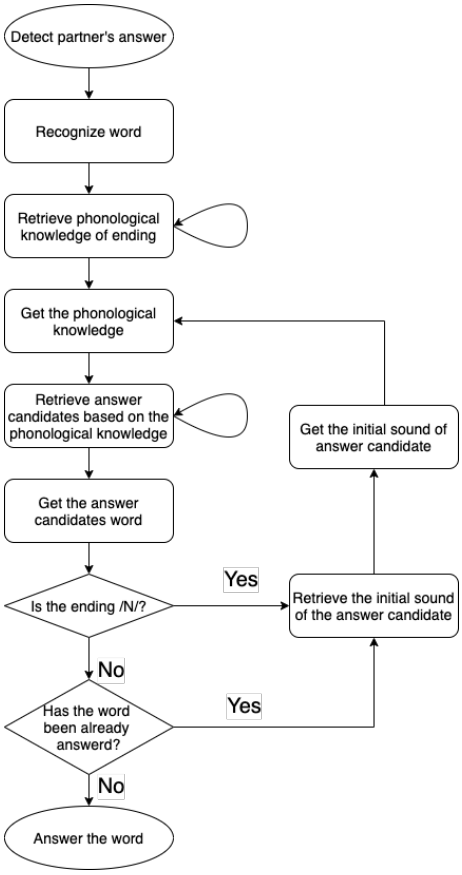


Figure 2: The answer process

(chunk type b in Table 2) is activated using this chunk. Activated phonological knowledge guides the retrieval of a chunk where the same mora is connected with a new word in an initial position (chunk type c in Table 2). When successful, the word knowledge in the retrieved chunk is stored in the goal module as an answer candidate.

After this, the model checks that the stored answer candidate is valid according to the rules of *shiritori*, such as not including /N/ as the end of a mora and not having been answered before in the current *shiritori* trial. If the current candidate violates these rules, the model re-searches for the answer candidate.

When the candidate word is confirmed as valid, the model stores it in the declarative module as an answered word and outputs the word through the speech module. During one *shiritori* task trial, the two agents alternately execute this procedure up to the given time limit.

Model Parameters

An important parameter in this study is related to chunk retrieval in the declarative module (activation). An activation is assigned to each chunk that is stored in the declarative module and affects the success or failure of the retrieval and the time required for the retrieval. The activity value A_i that is assigned to chunk i is defined as the addition of multiple terms as in Eq. 1:

$$A_i = B_i + S_i + P_i + \varepsilon_i \tag{1}$$

If more than one chunk matches the search request from the production module, the chunk with the highest activation is selected. In our model, among the activation elements, we focus on the similarity P_i and the base level B_i.

Similarity The similarity term P_i of the activation assigned to chunk i is computed using Eq. 2:

$$P_i = \sum_k P M_{ki} \tag{2}$$

This value is computed as the summation of the weighted degree of similarity M_{ki} for each retrieval request k to the chunk i. M_{ki} usually takes a negative value, and P serves as a penalty in the effect of similarity retrieval. Introducing similarity effects into the model's knowledge allows for the use of a mechanism called partial matching. For production module search requests, it is possible to search for chunks that do not have an exact match, thus allowing for flexible selection and the reproduction of certain errors.

In this study, for each phonological knowledge combination, we set M_{ki} based on the distinctive feature (Chomsky & Halle, 1968). That is, we assumed that the similarity between two phonemes can be defined as the overlap of distinctive features. In the simulation that is reported in the next section, we treat this parameter as the innate factor of language development, with the expectation of frequently observing errors involving the confusion of similar morae in the early phase of development.

Base Level The base level is the basic element of the activity value that corresponds to learning and forgetting. It is represented by Eq. 3:

$$B_i = \ln \left(\sum_{j=1}^{n} t_j^{-d} \right) + \beta_i \tag{3}$$

The value is computed from the number of presentations for chunk i (n) and the elapsed time since the chunk was referenced (t_j). d indicates the decay rate, and β_i is the offset parameter that can be modulated according to the simulation's aim. In this study, the base level is introduced into the model to examine how the experiential factor influences the development of phonological awareness. It is assumed that the innate factor's (similarity) relative importance decreases as base level activation increases and that errors related to phonological awareness are suppressed.

Learning Interactions

Children with inadequate phonological awareness need assistance from adults to play *shiritori* (Takahashi, 1997). In this study, we introduce assistance that encourages undeveloped agents to continue the *shiritori* task. We also assume asymmetric interactions, such as those between children and parents, setting the similarity parameter for only one of the two

agents. For the other agent, we incorporate the process of presenting the earlier answer again when the wrong answer has been received (for the wrong answer of *"o-ka-si"* (sweets) to *"ri-n-go"*, the agent presented *"ri-n-go"* again). Since ACT-R learning is calculated according to the frequency of using a chunk, it is expected that presenting correct answers increases activation of the correct association between words and morae.

Simulation

In this section, we describe three simulations using the model.

Error Occurrence Factors

Simulation Settings First, we test whether it is possible to model phonological errors using the ACT-R knowledge similarity function. Specifically, we perform simulations under the varying conditions of the ACT-R parameters (:mp), corresponding to the similarity weights P in Eq. 2.

The value of P was set to the following eight conditions: 1, 10, 20, 30, 40, 50, 60, and 70. Word knowledge in the model was selected from those listed in Amano and Kobayashi (2008)'s Japanese word database. Based on the rules of *shiritori*, we took out 20,544 nouns, excluding homonym duplications, and words consisting of only one mora, such as *"ro"* (furnace) and *"wa"* (ring). Next, referring to the child's associative vocabulary survey (for Japanese Language & Linguistics, 1981), 2,054 words were randomly taken out and used as model knowledge. For phonological knowledge, 103 pieces of knowledge were defined based on Japanese morae.

In this research, we use a unit called a "chain," which is the number of *shiritori* continuations. A trial in the simulation is terminated when the model achieves 100 chains or after 3,600 seconds have passed from the beginning of the trial. A total of 100 trials were simulated for each condition. Phonological knowledge similarity was calculated as the cosine similarity with the distinctive feature column of one mora as a vector, based on Chomsky's table of distinctive features (Chomsky & Halle, 1968).

Results and Discussion Figure 3 shows the types and numbers of errors that appeared during the simulation. In this graph, the horizontal axis indicates a phonological knowledge pair (morae) arranged in order from left to right based on similarity. The vertical axis shows the number of errors that occurred during the simulation using the pair of morae. For example, an incorrect answer of *"ri-n-go"* $\rightarrow$ *"o-ka-si"* counts as a pair of *"go – o."*

In the graph, the total number of errors is smaller in the condition where the value of P is larger, and the majority of errors is concentrated on the left side of the figure. From these results, it can be confirmed that phonological awareness errors are actually invoked by incorporating knowledge similarity. In addition, it can be observed that a larger similarity weight reduces the number of errors, and that errors are less likely to occur in low-similarity pairs. This is thought to correspond to the phenomenon that was confirmed in the child development process where the *"ra"* sound is confused with a *"da"* sound (Kobayashi, 2018).

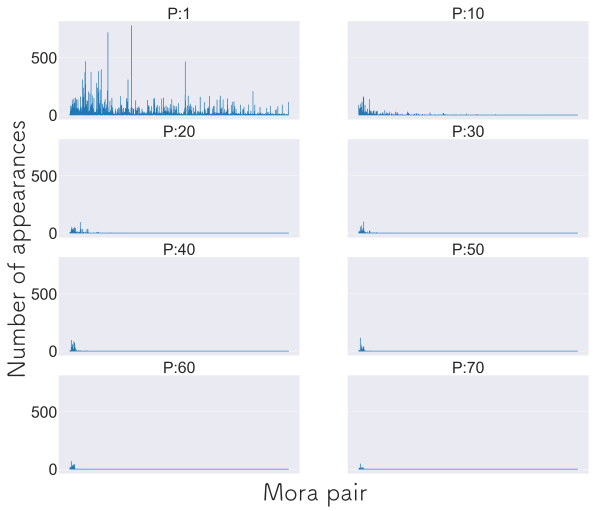


Figure 3: A comparison of similarity differences and errors

Error Suppression Factors

Simulation Settings Following the simulations that were described in the previous section, we will now examine the factors that suppress phonological errors. Here, we hypothesize that the impact of similarity is relatively reduced by the effect of learning. To test this hypothesis, we simulate a condition in which the ACT-R parameter (:blc) corresponding to the value of the offset β_i in Eq. 3 is changed to the following six conditions: 0.1, 1, 5, 10, 15, and 20, while keeping the same setting in the other parameters, as in the previous section's simulation.

In addition, we examine whether learning occurs as a result of the execution of the tasks that were set in this study. We set the condition that the offset β_i is set to 0 and observe the time-series change through the execution of the task.

Results and Discussion Figure 4 shows the change in the rate of correct answers due to the manipulation of the offset value β_i in Eq. 3. The horizontal and vertical axes, respectively, represent the value of β_i and the rate of correct answers defined as the ratio of words suitable for the *shiritori* rule from among all the words that were provided during the task. The graph shows an increase in correct answers as the base level constant β_i increases, suggesting that the effect of learning B_i increased and the similarity effect P_i decreased relatively with respect to the ACT-R activation calculation (Eq. 1). In other words, the result indicates that the learning effect can suppress the phonological errors that the similarity effect causes.

Figure 5 shows the change in the rate of correct answers given during the execution of the task. The horizontal axis shows the time spent completing the task, divided into four intervals, while the vertical axis shows the percentage of cor-

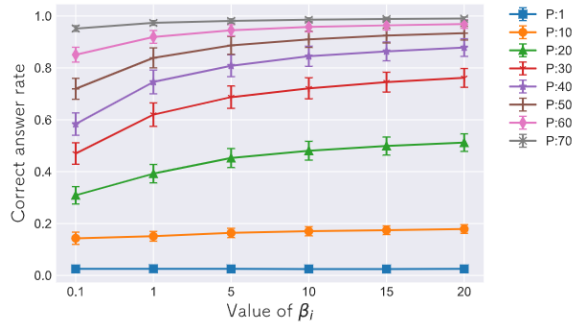


Figure 4: The change in the correct answer rate, as indicated by β_i (error bar is $\pm\frac{SD}{5}$)

rect answers in each interval. In the graph, there is no condition showing that the rate of correct answers is increasing, in some conditions, it is decreasing. This indicates that learning does not occur in the task that was set in this study.

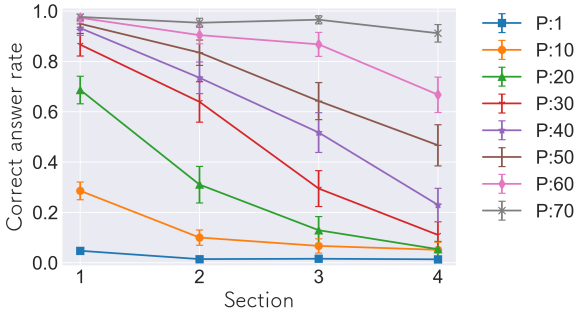


Figure 5: Changes in the rate of correct answers throughout the task (error bar is $\pm\frac{SD}{5}$)

In this regard, it is suggested that implemented assistance (repeating the previous answer) has no effect on learning, and we need to explore other forms of support to enhance phonological awareness in the *shiritori* task. In Takahashi (1997)'s study, the most effective assistance came in the form of offering hints that activate semantic knowledge pertaining to the correct answer.

In addition, through individual case observation, the number of times that *shiritori* has continued stands at about 10, and it is possible that a sufficient number of presentations of knowledge for learning may not have been secured. Furthermore, due to the research process' effect with regard to avoiding words that were already provided as answers and words that end with /N/, a problem can be assumed in which only the same words are searched and it is impossible to proceed to the next answer.

Summary and Future Work

The purpose of this study was to investigate the factors that contribute to the occurrence and suppression of errors during the formation of phonological awareness. For this purpose, we modeled phonological awareness using the cognitive architecture ACT-R and performed simulations that manipulated ACT-R parameters corresponding to both innate and experiential factors. As a result, it is suggested that phonological awareness and the errors that occur during its formation process can be modeled through the innate memory retrieval mechanism. Furthermore, the effect of learning may reduce errors by suppressing innate factors. However, the results did not show that learning occurred while performing the tasks set in this study.

This study leaves a number of issues to be addressed. In its simulations, the authors arbitrarily determined values for each parameter. Although we were able to observe a variation in the correct answer rate, the threshold and maximum values need to be verified using further simulation. In addition, this study only showed that the correct answer rate was improved by setting the parameters in advance; it did not show that learning occurred or that the correct answer rate was improved through task performance. It is therefore necessary to further examine the interaction in which the learning effect can be expected to operate. Consideration should also be given to non-phonological aids, such as letters, pictures, and word meanings. In addition, evaluating the constructed model is essential. To ensure that the model's task-related learning process explains the child's language development, it is crucial to correlate it with experimental data from humans. We believe that after setting the learning task, it is possible to examine this by comparing it with the existing human experimental data (Hara, 2001).

Acknowledgment

This research was supported by JSPS KAKENHI Grant Numbers 18H05068, 20H05560, 20H04996.

References

Amano, S., & Kobayashi, T. (2008). *Kihongo database : gogibetsu tangoshimmitsudo.* Gakken plus.

Anderson, J. R. (2007). How can the human mind occur in the physical universe?

Baron-Cohen, S. (1997). *Mindblindness: An essay on autism and theory of mind*. MIT press.

Bloch, B. (1950). Studies in colloquial japanese iv phonemics. *Language*, *26*(1), 86–125.

Chomsky, N., & Halle, M. (1968). *The sound pattern of english*. Harper & Row New York.

for Japanese Language, N. I., & Linguistics. (1981). *Yoji / jido no rensogoihyo*. Tokyo Shoseki.

Hara, K. (2001). Kenjoji ni okeru oninishiki no hattatsu. *The Japanese journal of communication disorders*, *18*(1), 10–18.

Ishida, H., & Ishizaka, I. (Eds.). (2016). *Gengochokakushi no tameno gengohattatsushogaigaku*. Ishiyaku shuppan.

Kobayashi, H. (2018). *Oninishiki no keisei to kotoba no hattatsu -"kotoba ga osoi" wo kangaeru-*. Kodama shuppan.

Kubozono, H. (1989). The mora and syllable structure in japanese: Evidence from speech errors. *Language and Speech*, *32*(3), 249–278.

Kubozono, H. (2000). Kodomo no shiritori to mora no kakutoku. *Bulletin of the Faculty of Letters, Kobe University*(27), 587–602.

Mätzig, P., Vasishth, S., Engelmann, F., Caplan, D., & Burchert, F. (2018). A computational investigation of sources of variability in sentence comprehension difficulty in aphasia. *Topics in cognitive science*, *10*(1), 161–174.

Mugitani, R., Homae, F., Hiroya, S., Satou, Y., Shirose, A., Tanaka, A., ... Tachiiri, H. (2019). *Kodomo no onsei* (No. 21). Corona Publishing Co., Ltd.

Nakamura, T., Kojima, C., & Fujiwara, Y. (2015). Kenjohattatsu ni okeru onin purosesu no henka. *Journal of rehabilitation sciences, Seirei Christopher University*, *10*, 1–13.

Stahl, S. A., & Murray, B. A. (1994). Defining phonological awareness and its relationship to early reading. *Journal of educational Psychology*, *86*(2), 221.

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say "broke"? a model of learning the past tense without feedback. *Cognition*, *86*(2), 123–155.

Takahashi, N. (1997). Yoji no kotobaasobi no hattatsu : "shiritori" o kanon isuru joken no bunseki. *The Japanese Journal of Developmental Psychology*, *8*(1), 42–52.

Tomasello, M. (1999). *The cultural origins of human cognition*. Harvard University Press.

Trubetzkoy, N. S. (1969). Principles of phonology.

Van Rij, J., Van Rijn, H., & Hendriks, P. (2010). Cognitive architectures and language acquisition: A case study in pronoun comprehension. *Journal of Child Language*, *37*(3), 731–766.

A Cognitive Model of Sound Representations in Children with Speech Sound Disorders

Soujanya Pathi (la17resch01002@iith.ac.in)

Prakash Mondal (prakashmondal@la.ith.ac.in)

Department of Liberal Arts

Indian Institute of Technology Hyderabad

Abstract

The goal of the current work is to develop a theoretical model that can possibly account for certain speech disarticulations that occur among children with Speech Sound Disorders (SSDs). In trying to do so, we propose an interface module, a specialized transducing system within the speech sound system, the nature and functioning of which in the cognitive system may provide us with some useful insights into SSDs. The postulation of an interface module here is necessitated by the fact that there are facets of errors in SSDs and in typical populations that cannot be simply explained in terms of either articulatory/phonetic factors or matters pertaining to abstract sound representations. This paper will, therefore, present a detailed view of the interface, its nature, its relation to levels in the cognitive system, and the functions it performs. The results of applying the proposed model to certain types of sound alterations in SSDs are described with implications for the cognitive representation of speech sounds.

Keywords: Speech Sound Disorders; Phonological Representations; Interface.

Introduction

The realization that language is fundamentally a complex system has helped understand the significance of the human mind as a cognitive system. In this context, the analysis of speech production has played a crucial role in first formulating cognitive theories and then subsequently applying them in the real-world context. However, there are certain gaps of understanding in speech production, specifically in speech sound disorders, that elude satisfactory explanations. Speech language pathology has, of course, made several advancements in terms of speech diagnosis and treatment. Nonetheless, there is more to the issue than meets the eye. More often than not, speech language pathology has catered to speech disorders that relate either to articulatory disorders or to matters pertaining to phonological representations (PRs). There are, however, certain facets of SSDs that cannot be simply explained by either. How do we treat or possibly explain the speech dysfunctionalities of a child, for instance, who displays no motor or structural abnormalities, and also at the same time demonstrates a capacity to discriminate two different sounds? While physiological defects are entirely ruled out in such cases, one cannot account for such speech errors by merely appealing to phonological representations.

If PRs, as popularly propounded by many (Dodd, 2005; McNeill & Hesketh, 2010; Anthony et al. 2011, Sutherland & Gillon, 2007), are to be held responsible for the disarticulation of the sounds, then it simply does not explain why and how a child can possess the capacity to distinguish a minimal pair set. Likewise, if we are to advance the inaccessibility of the mentally instantiated phonological symbol as the reason for speech dysfunctionalities, it does not always seem reasonable to attribute all the speech problems to a loss of a certain cognitive capacity that cannot, for reasons that are not very clear, access the right symbol from the phonological system. Moreover, given the understanding that speech disorders mostly exhibit a pattern in terms of the errors they commit, the symbol extraction problem does not seem to spell out the exact reasons as to why only certain specific sounds (under predictable environments) are misarticulated. It is cases like these that we believe require closer inspection and probably more explanation in terms of what kind of cognitive processes drive a child to produce a certain sound in a way that is deviant from the typical speech.

Thus, this paper outlines a theoretical model that focuses on contributing to an understanding of the internal cognitive mechanisms/procedures that ultimately lead to variants in speech production in typical and atypical populations. For the same, we will discuss in detail, with the help of relevant data, how a model such as ours can account for variations in speech differences.

Cognition and Speech Disorders

Speech sound disorders are speech and language disorders identified by the inappropriate use of speech utterances, which may involve errors in the production, perception or organization of speech sounds. These are particularly relevant to the present study because the current work intends to look at the cognitive processes of the speech sound system not in terms of how effectively the speech system functions in producing the correct speech utterances, but in terms of how inadequately the system can possibly work in the production of unintended utterances. Numerous different categories of models such as the connectionist model of Dell, Change & Griffin (1999), Fromkin's five-stage model (Fromkin, 1971) and Garrett's model (Garrett, 1975) have been developed in an attempt to account for speech sounds in general. Our study, however, differs in its effort to consider the disordered data as its preliminary basis to explicate the cognitive procedures in the speech sound system, and thus emphasizes the need for an 'interface' module inside the human cognitive system in view of the specific patterns of errors found in speech sound disorders.

The relation between cognition and disordered speech has been a subject of investigation for many years, and the results have only cemented the already existing belief that they exert a rather strong influence over each other. Shriberg & Widder's (1990) findings from nearly four decades of speech research in cognitive impairment indicated that persons with cognitive impairments or any

sort of deficits at a cognitive level are likely to have speech problems. That is, the articulatory skills of a subgroup with cognitive deficits differed significantly from the normally developing children. Similarly, in several of the case studies conducted by Sutherland (2006) on children with severe speech impairments, it was observed that 3 out of 4 children demonstrated poor phonological skills. The results indicate that children with consistent speech impairments experienced deficits at the cognitive-linguistic level (i.e., phonological representations) of speech production (Dodd, 2005). Cognition and disordered speech are inexplicably intertwined in a manner that is overt in terms of the influence the former exerts over the latter, and yet imprecise in terms of the exact cognitive processes or the mechanisms that actually result in specific speech dysfunctionalities.

The purpose of this study, therefore, is to address these crucial gaps and explore in some detail the phonological processes in the mind that can be held accountable for some of the dysfunctional speech. We will discuss in detail the phonological processing of speech sounds in both typical and atypical populations in the later sections. Before delving deep into the confinements of the modelling theory, we shall first have a look at the PRs.

Phonological Representations

The concept of a phonological representation has undergone a significant number of changes over the period, and there is no single classification that can possibly take into consideration the difference of opinions that had gone into defining PR in specific frameworks. However, there is a common consensus among scholars that PRs represent the underlying structures of the sound system. Locke & Studdert-Kennedy (1983) formally define phonological representations as the underlying sound structure of specific words stored in long-term memory. Berent (2013) considers PRs to be discrete and combinatorial. These representations reportedly distinguish different kinds of symbols that are instances of some form in the real or imaginative world. Berent also emphasizes a categorical distinction of different classes of sounds as a means to form a specific category of members that are alike. Such a perception of PRs seemingly supports generalizations that apply across the board to all members of a class. A large amount of relevant literature on the study of PRs, however, also differ significantly on account of numerous parameters contributing to PRs. While Browman & Goldstein (1986) describe PRs based on articulatory gestures, Halle (1985) associates them with certain mental scripts. Foley (1977), on the other hand, believes that PRs denote abstract, phonological objects-a proposition, which of all other assumptions somewhat resonates with the view adopted in our current study. Coleman's (1998) view on phonological representations (bearing acoustic signatures of sounds) as something capable of supporting computations of some kind also holds importance in our present-day analysis of sounds.

As far as the presentation of PR in our present analysis is concerned, our study adopts the concept of an element in accordance with the Element theory (Kaye & Harris, 1990; Harris, 1994; Backley, 2011), which was developed as part of Government Theory. The need for a PR deviant from a traditionalistic view is necessitated by the latter's inability to explain certain facets of SSDs that simply cannot be explained by considering PR as a system that has within itself all the phonetic details of the sounds. As mentioned in the introduction, there are cases that cannot be simply explained by appealing to the PR. A case in point is a study conducted by Leahy & Dodd (1987) where it was revealed that a child with bizarre phonological processes such as the deletion of final consonants or marking them by a glottal stop, and no apparent articulatory defects produced defective sentences. Despite her ability to discriminate minimal pairs, or to recognize her own errors as errors, the subject in question exhibited abnormally deviant patterns of speech. The data demonstrated that there was no deficit in perceptual processing, indicating that there was no apparent problem with the PR as well. In cases such as this, it becomes necessary that we revise the notion of PR that can maximally account for most of the speech sounds, if not all.

In the present analysis, PR is merely viewed as a system that contains certain elements, the combination of which gives rise to more complex segments. It consists of a set of finite elements like |A|, |I|, |U| that correspond to different acoustic properties of speech sounds. To elaborate, let us take two elements |X| and |Y| that combine to give rise to a segment say z. The resultant segment z, formed by the combined acoustic properties of two or more elements, is fed into a system where it takes another form. For instance, the element |A| corresponds to central spectral energy mass where high F1 converges with F2, and the element |I| corresponds to high spectral peak where high F2 converges with F3. The physical correlates of the element |A| roughly represent the gutturals (e.g. pharyngeals, uvulars) and some types of coronals, while |I| represents palatals and other types of coronals (Backley, 2011). Likewise, the segment /æ/, which imbibes the acoustic properties of both |A| and |I|, can be viewed as the combination of both the elements.

Although the elements in PR tell the speakers which patterns they must produce, it does not tell the speaker how to produce them. The description of the sounds in terms of the vocal tract and the ways in which it constrains the production of speech sounds physically are all realized at a level beyond the PR. For now, the function of the PR is to simply provide the underspecified inputs to the system where the segments can be further processed.

What contributes to a deficiency in speaking and reading is often attributed to primarily having deficits at the level of PR. Most of the research claims that the children with speech impairments produce erroneous segments or sounds because their PR, by default, is disturbed (Bernthal, Bankson & Flipsen, 2009; Anthony et al., 2011; Johnson, Pennington, Lowenstein, and Nittrouer, 2011; Sutherland and Gillon, 2005, 2007). That is to say that the PR of a person with a speech disability and that of a person without a disability are different. In other words, it seems to suggest that the disabled have a defective PR and the abled a perfect PR. Nevertheless, more often than not, we see that linguistic phenomena like metathesis and spoonerisms are not uncommon in persons without speech disorders. While one can argue that they are mere 'slips-of-the- tongue' and therefore correspond to the articulatory factors than to any representational factors, it is also worth noting that these 'slips- of- the-tongue' also often provide useful insights into the phonological structure of the language (Fromkin, 1971; Harrikari, 1999). It is therefore suggested that it is not viable to directly dismiss or establish PR as being either imperfect in the case of SSDs, or totally perfect as in the case of persons without speech impairments. Rather, what seems to be more plausible is to view PR as a representational system that is just 'good enough' in both cases. The correct or the incorrect utterances produced at the articulatory system are not because of 'mental misrepresentations' at the PR level

but somewhere else. We propose that there is an intermediate system in the cognitive system, somewhere between the PR and the articulatory system, where a variety of operations take place. We presume any malfunction at this stage can possibly lead to speech errors. We call this intermediate system an interface whose nature and functioning is discussed in the next section.

The Interface System

The concept of an interface is operationally defined in this paper as a transducing system that receives, alters, processes, and shares the phonological information from one level to the other. The notion of the interface is primarily proposed to account for the sensitivity to points of intersections between a system of PR and the articulatory system where the idea of autonomy is quite constrained. Though Ohala (1990) argues against using the term 'interface' between a system of phonology and phonetics, we choose to preserve the term for reasons which we will discuss in brief. Firstly, Ohala's objection is that an interface translates itself into a mechanism that passes the information from the system of phonology to phonetics when the two are assumed to be mostly independent of each other. Explicitly discussing the concept of an interface in the context of phonology and phonetics, Ohala contends that phonetics and phonology, for several reasons, are two domains of the same speech universe that are highly intertwined with each other. Therefore, the use of the term interface is inappropriate as it leads people to believe that the two domains are mostly autonomous, bearing very little or no interaction on each other. This work, however, chooses to use the term to designate a system in itself in line with Jackendoff's (2002) concept of interface systems, even though it does not advocate the outdated view that the system of phonology and phonetics are autonomous, as Ohala also thinks. In fact, the interface in the present context is viewed as a system that mediates the link between the two domains. We draw upon the concept of an interface as a system that does not mediate between two different domains of language but functions within the narrow spectrum of phonology. Therefore, we suggest that there are different levels within the mentally instantiated phonological system, one of which is the interface.

Thus, in line with Fodor's (1983) conceptualization of 'domain specificity', the interface system is domain-specific in being restricted to only classes of phonological objects, and the information processed in the system (interface) is circumscribed in a comparatively narrow way. Though the proposed interface is *autonomous* in terms of its functionality and nature, it also at the same time allows for the flow of information into its mechanisms from the system of PR conceived of as an inventory of elements and their combinations. The interface is autonomous with respect to syntax and semantics, for example, only in the sense that its intrinsic functioning is not affected by the contents and operations in syntax and semantics. While the interface maintains domain-specificity, the interface system is *informationally encapsulated* relative to syntax and semantics, for example, because only phonological objects and the internal grammar for operations on such objects are relied on. However, the interface system is not *informationally encapsulated* relative to the subsystems of the entire speech sound system because it interacts with the system of PR. As for the neural architecture, the interface may be instantiated by Poeppel's (2014) *sensory-motor interface* and *phono- logical network* involving the left Sylvian parietal-temporal fissure and the bilateral superior temporal sulcus. Poeppel's *articulatory network* may not be involved because that is responsible for sound articulation, while the interface is not directly responsible for articulation

The interface system in its functioning significantly differs from Dell's connectionist model of spreading activation (Dell, 1986) because, in Dell's model, there is no scope for the mapping of the symbolic units of phonology to articulatory instructions. Besides, phonological units are not decomposed into their components-acoustic or otherwise-although the significance of phonemes is acknowledged (Dell, 2014). The idea of an interface of a similar kind has been discussed in detail by Reiss and Volenec (2017), where the interface is seen as establishing an intermediate level between phonology and phonetics. The paper's goal jells with the central idea of our line of work in terms of positing an interface that operates on its own set of rules.

Additionally, Reiss and Volenec's work renders the transduced features $PR_{[ROUND]}$ or $PR_{[+BACK]}$ in terms of the muscular contractions that each of them relates to. It is, however, still unclear as to what exactly triggers the articulatory movements for the specific sounds. The authors also, of course, discuss how these temporally coordinated muscles are related to features, but it is not clear by what means the feature information in the sensory-motor gets translated into the actual rounding of the lips (in the case of $PR_{[ROUND]}$) or the real-time function of raising the back of the tongue (in the case of $PR_{[+BACK]}$). In other words, it is not clearly established what articulatory aspects get encoded in each specific feature and how these features interact with specific muscles. However, for the purpose of speech externalization, there has to be a mechanism that explicitly states or provides at least some kind of a signal for the articulatory movements to get started. In trying to address this key issue, the current work has adopted a model of an interface system that not only bridges the gap between the abstract mental representations of the sounds and the actualizations of the sounds, but also specifies in a series of steps what exact instructions have to be followed in order to produce a particular sound.

In spite of an indication in our theoretical model that the interface is linked to other domains of language like syntax and morphology since no sound can be produced in isolation, the study of the interaction between the interface and the other domains is outside the purview of the present study. Henceforth, the nature and the functioning of the interface will be presented as it occurs in a relatively narrower spectrum of the phonological system.

How does the interface system work?

The input received from the PR is fed to the interface module where certain operations take place. The interface module can be considered as a workspace that hosts a set of slots comprising of articulatory features pertaining to both consonants and vowels. Depending on the phonology of each person, these slots are filled up with relevant articulatory features (See figure 1). We suggest that it is usually the mapping or rather the mis-mapping of the input segment from the PR to the relevant slots that produce a defective speech.

While the gestural movements of the articulators in different combinations primarily indicate the formation of a particular sound, they also explain the human tendency to prefer specific sound patterns over others. For instance, John Ohala (1983) argues for the absence of the /g/ sound over the other voiced sounds in terms of the aerodynamic factors. He maintains that the sound /g/ is more susceptible to deletion than any other voiced plosive by the degree of its closeness between the larynx and its point of closure. Because the location of the closure is much closer to the larynx, the air pressure in the supraglottal region exceeds that of the air below the larynx, thereby leaving insufficient air to drive the vibration in the vocal cords. Similarly, we suspect that a predisposition to using certain sounds over others can be traced back to the mapping of the input segments from the PR, which can be aerodynamically motivated. While most of the errors analyzed in the model are errors due to the mapping problem as part of the interface, but these errors are not random and hence they can arise as epiphenomena or 'side effects' of the ongoings within the articulatory system. Because PR is solely a representative module devoid of articulatory slots, we presume that the articulatory or phonetic factors are *partly* instantiated in the interface (in slots) and *wholly* manifested in the articulatory system. Consequently, a mis-mapping of the underspecified input from the PR onto the wrong slot evinces the instantiation of an unintended property of a specific sound, thereby resulting in the production of a disordered utterance. This kind of analysis is particularly helpful in analyzing the sound patterns in persons with SSDs since most of the errors produced in SSDs, if not all, dovetail with patterns indicating a preference for one sound or one class of sounds over others.

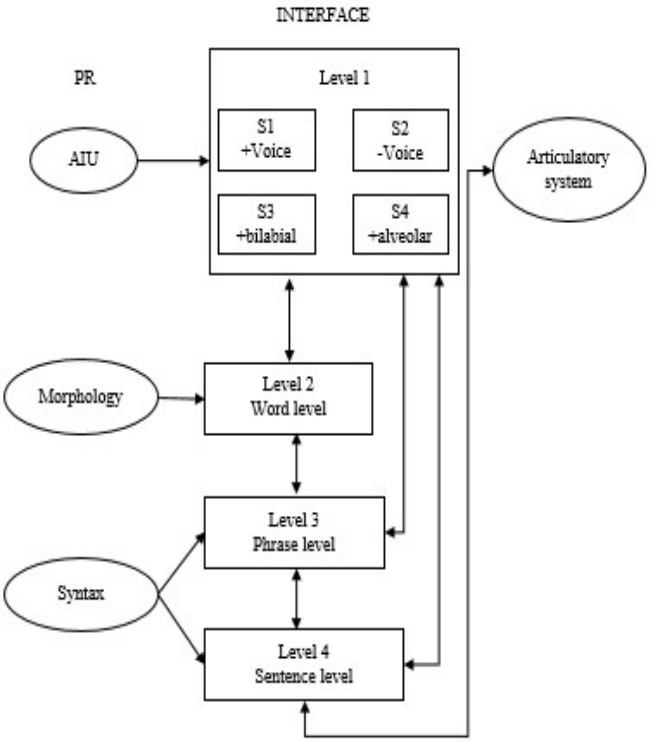


Figure 1: The model comprising of the PR, the interface and the articulatory system.

Samples of Data

Case 1

Presented in table 1 is the clinical case conducted by Barlow and & Gierut (2002) on a 4-year old child Joseph who was diagnosed with functional phonological delay. The child displayed a variety of speech errors, a few of which are currently drawn from the large-scale study to illustrate how our proposed theoretical model can accommodate the actual data. The child in question displayed normal hearing, intelligence, oral-motor functioning, and regular receptive and expressive language skills as per the formal testing procedures. Joseph's speech data display several gaps in terms of the normalized phonetic inventory of the English language and some deviant patterns that are otherwise not to be found. The kind of errors ranged from simple substitutions and deletions to cluster simplifications or a combination of all of these. Figure 2 demonstrates level 1 of the interface in terms of how the substituted /t/ (/ tʌnɪ /) for /s/ (sunny) can be articulated, followed by their set of operations.

Table 1: Joseph's data

S.No	Target	Utterance	S.No	Target	Utterance
1	bite	bɑɪ	9	kids	kɪp
2	bus	bʌ	10	mud	mʌ
3	cheese	tɪ	11	tooth	tʊʔ
4	cut	kʌʔ	12	drive	gɑɪ
5	five	pɑɪ	13	sharp	tɑʊp
6	gift	gɪp	14	soap	to
7	toes	to	15	sunny	tʌnɪ
8	juice	dʊ	16	soup	tʊʔ

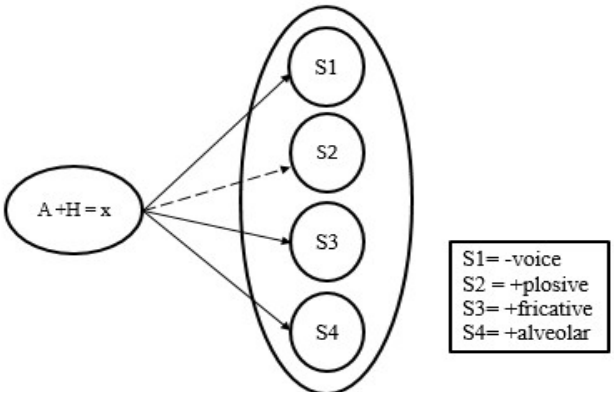


Figure 2: Straight line = correct mapping, dotted line= incorrect mapping.

Operations

For the intended utterance /s/,

Step 1: A+H = x, where x is the underspecified segment from PR

Step 2: x+S1+S3+S4 = initiation of /s/ sound

For the disordered utterance /t/,

Step 1: A+H = x, where x is the underspecified segment from PR
Step 2: x+S1+S2+S4 = initiation of /t/ sound

While the sounds /s/ and /t/ differ minimally on a single slot, they also share the same place of articulation and voice and yet the mis-mapping of one single sound may result in a collapse of contrast between two sounds. As was also seen in Joseph's case, the sound /s/ never occurred in his phonemic inventory of sounds. Hence we can possibly infer that the mapping, or rather the mis-mapping of the S2 slot from the underspecified PR segment, by way of fossilization, has permanently been established. The presence of the articulatory features and the mishaps in the operations performed at the level of the interface also serve as an explanation as to why Joseph's receptive skills are still intact, despite his inability to produce the sounds correctly. Because the present model considers PR to be an efficient system with almost no malfunctions within it, we assume, Joseph still displays the capacity to understand /s/ and /t/ as two distinct sounds.

Case2

In case 1, we have looked at errors of substitutions and their operations at the interface level. We will now look at how deletions can possibly be explained by the model. For that purpose, we will consider another set of sample data from a case study conducted on a subject named Josie between the ages of two and five (Bowen, 2015). Josie was diagnosed with developmental verbal dyspraxia (DVD) and had performed poorly on articulatory tests. Her speech was rendered unintelligible despite maintaining a mid-range receptive, expressive, and total language score. The data used in table 2 are impoverished and a part of the sample prior to the intervention.

Table 2: Josie's data

S.No	Target	Utterance	S.No	Target	Utterance
1	cup	kʰʌ	11	snake	fneɪʔ
2	gone	kʰɒn	12	house	hæʊ
3	Knife	nɑɪ	13	toe	tʰoʊ
4	sharp	wja:	14	mouth	maʊ
5	fish	de	15	nose	noʊ

Josie's disorder was severe and often exhibited patterns that were most likely unintelligible. Though intervention studies altered Josie's speech at a later stage, for the purpose of our study, we shall first try to investigate what, in the first place, had caused such chronic distortions. Josie exhibited a range of patterns starting from single sound substitutions and deletion to the production of sounds that bore no resemblance to the target word. One possible explanation for the case of deleted sounds could be traced back to the inactivity in the slots. That is, there could be instances when the slots do not function actively even in the cases when they are required to do so. The inactivity of a slot can eventually lead to two consequences: firstly, the segment generated from the PR, upon finding the slot inactive, deviates to other slots, thereby producing a different segment. So far, this mis-mapping has served as an explanation for the substituted sounds. Secondly, the segment generated from the PR, upon finding the slot inactive or invalid, does not end up being assigned any feature. However, in this case, the segment does not get itself 'attracted' to the wrong slot. Instead, the segment is left in situ, devoid of any articulatory features to process. Specifically concentrating on the case of /p/ deletion in the word 'cup', we speculate that the slots holding the corresponding features of /p/ fail to assign the articulatory features to the segment generated from the PR. Moreover, because the segment has been assigned a null value, no particular articulatory instruction is taken forward for the next levels. As a result of this, there is no production of the sound /p/ in the articulatory system. The transitory nature of the slots also justifies why certain slots holding features like –voice remain passive in the production of /p/ but stay active in the production of other voiceless sounds like in the production of /f/. It is plausible that certain slots can go inactive for certain element combinations in this way due to the impact of relevant aerodynamic factors, as discussed in the previous section. Hence, it is essentially the nature of the slots that result in the deletion and not the mapping. Illustrated in figures 3a and 3b are the inactive slots in /p/, and the active slots in /f/ respectively.

Significantly, the current model can capture other datasets, two of which are briefly discussed here. The first case comes from the numerous studies conducted on pre-school children with speech impairments (Sutherland & Gillon, 2005). The data involve a wide range of substitution errors in both vowels and consonants. While there are some observable patterns in terms of which syllables (stressed or unstressed) or segments (either consonants or vowels) were generally prone to disarticulation in the children, the quality of the sounds produced is highly impoverished. The second is a case history of a 3-year-old child Kirk (Bernthal, Bankson, & Flipsen, 2017) who exhibits poor intelligibility in speaking despite having normal motor and language development. The child displays unusual processes like initial consonant deletions or final constant deletions, which are atypical for a 3- year-old child. As far as the substitution errors are concerned, the analysis has revealed that stopping was the most dominant and the most preferred process of all. With /d/ substituting the likes of /f/, /v/, /θ/, /d/, /s/, /z/, /ʃ/, /tʃ/, and /dʒ/, the sound emerged as the most prominent sound in Kirk's vocabulary. A similar pattern was also observed in Joseph's case, where the child had also exhibited a similar preference for the use of plosives instead of fricatives and affricates.

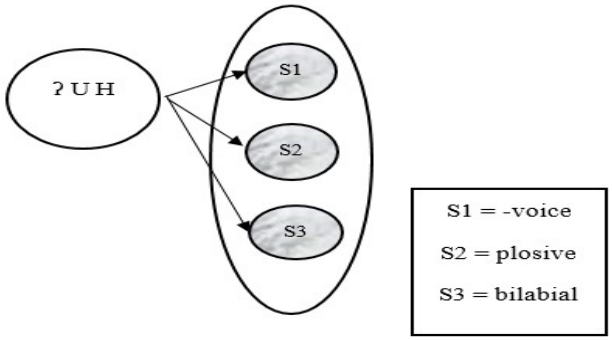


Figure 3a: The inactive slots in /p/ indicated by the marbled slots.

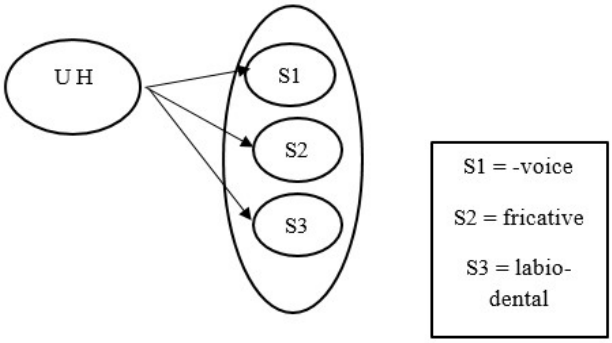


Figure 3b. The active slots in /f/ indicated by the plain slots.

Beyond Level 1 of the Interface

Though there are no coarticulation, assimilation, or preservative errors to be found in the present data, our present model can in fact account for those types of errors, if necessary. The levels within the interface system are not sub-modules that are strictly compartmentalized but are presented in a fashion such that they feed information, albeit in constrained ways, into each other. Level 1, which happens to the most important levels of all (because the initiation for the primary articulation takes place here) is connected to all the other levels in a bidirectional way. The word level, phrasal level, and the morpheme level are all again connected to each other in a bidirectional way. To illustrate how this functions, let us take the example of the production of sound /f/. Let the elements U and H combine to form an underspecified segment say x. 'x' is then rightly associated with the slot which provides the feature -voice. The sound thus generated passes on to level 2 (the word level), where it checks for the neighboring sounds. If there is a possibility of the segment getting altered, as in the case of coarticulation, it reverts to level 1 (since we mention that it is a bidirectional system) and picks up the required slot. The newly generated sound is again sent to the word level and further on. If the neighboring sounds do not affect the sound in any way, then they simply get carried on to the next level. The set of operations that occur at this word level can be schematized as follows:

Case 1: If f→f, then move to level 3.
Case 2: Step 1; If f→x, then revert to level 1, where x is the new modified sound
Step 2; Select new slot
Step 3; Generate required sound
Step 4; Pass through level 2

Similarly, the shifts in sounds at the phrasal level and the sentential level can also be explained by connecting them to level 1, both in disordered and typical speech.

Implications

The representative errors in SSD complexities, specifically those concerning the sound structure, stem from either 'misrepresented symbols' or from various processing deficits. Therefore, in order to have differential diagnosis and treatment therapies for the SSDs, the SSD classification must be efficiently established. Based on the earlier developments, and the current advances in neurolinguistics, several systems of SSD classifications have been proposed, some of which have had implications for the differential diagnosis and treatment planning. (Waring and Knight, 2013; Shriberg et al., 2010; Dodd, 2014; Stackhouse and Wells, 1997). However, these classifications, as Terband et al. (2019) claim, do not thoroughly explore the relationships between the different levels of causation, and hence, may deter efficient diagnosis, customize intervention, and optimize outcomes. The present cognitive model explores different levels in the speech sound system and thereby identify the 'cause' of the speech deficit. The implications of such a model extend to an advantage in recognizing SSD subtypes that evince a perfect PR and yet a defective speech output. The ability of SSD patients to identify and discriminate phonemes in relation to their ability to produce sounds, measured on standard clinical diagnostic tests, for instance, serves as a good predictor of the PR efficiency. Thus, an experimental validation of the model, which is beyond the scope of this paper, can be fruitful.

Further study

Owing to the dearth of data that specifically relate to the mental representations of sounds, and to cases where the PR is presumably good enough, this study has been able to look at only few issues. Firstly, this study does not take into account other levels of speech errors occurring at syllabic and discourse levels. It is hoped that a segmental view at first would provide the appropriate information which would, in the future, contribute to accounts for the phrase or discourse level as well. Secondly, this study has looked only at the word substitutions deletions and no other frequently occurring linguistic phenomena such as transposition at a segmental level. The inclusion of other linguistic phenomena such as transposition would require further modifications to the present model.

Conclusion

The present study has attempted to explain why certain clinically notable segmental speech errors, which cannot be explained by significant impairments in the motor, sensory, or even mental representations of sounds, occur in cases of SSDs. The concept of an interface module has been advanced with possible miscalculations at any level of coding resulting in an inaccurate or an unintended utterance. As of now, claims cannot be made if this particular model suffices for all kinds of segmental errors, but further inquiry can help fine-tune the present model further.

References

Anthony, J. L., Aghara, R. G., Dunkelberger, M. J., Anthony, T. I., Williams, J. M., & Zhang, Z. (2011). What factors place children with speech sound disorders at risk for reading problems? *American Journal of Speech-Language Pathology*, *20* (2), 146–160.

Backley, P. (2011). *An Introduction to element theory*. Edinburgh: Edinburgh University Press.

Barlow, J. A., & Gierut, J. A. (2002). Minimal pair approaches to phonological remediation. *Seminars in Speech and Language*, *23* (1), 057-068.

Berent, I. (2013). *The phonological mind*. Cambridge: Cambridge University Press.

Bernthal, J. E., Bankson, N. W., & Flipsen, P. (2009). *Articulation and phonological disorders: Speech sound disorders in children.* Boston, Mass: Pearson/Allyn & Bacon.

Bowen, C. (2015). *Children's speech sound disorders*. Chichester, West Sussex: John Wiley & Sons.

Browman, C. P., & Goldstein, I. (1986). Towards an articulatory phonology. *Phonology Yearbook, 3,* 219-252.

Coleman, J. (1998). *Phonological representations*. Cambridge: Cambridge University Press.

Dell, G. S. (1986). A spreading activation theory of retrieval in language production. *Psychological Review, 93*, 283–321.

Dell, G.S., Change, F., and Griffin, Z.M. (1999). Connectionist models of language production: Lexical access and grammatical encoding. *Cognitive Review, 23*, 517-542.

Dell, G. S. (2014). Phonemes and production. *Language and Cognitive Process*, 29 (1), 30-32.

Dodd, B. (Ed.). (2005). *Diffrential diagnosis and treatment of children with speech disorder.* London: Whurr Publications.

Dodd, B. (2014). Differential diagnosis of pediatric speech sound disorder. *Current Developmental Disorders Reports, 1*(3), 189-196.

Fodor, J. A. (1983). *The modularity of mind*. Cambridge, MA: MIT Press.

Foley, J. (1977). *Foundations of theoretical phonology.* Cambridge: Cambridge University Press.

Fromkin, V, A. (1971). The non-anomalous nature of anomalous utterances. *Language*, 47, 27–52.

Garrett, M. F. (1975). The analysis of sentence production. In G. Bower (Ed.), *Psychology of learning and motivation,* 133-175. New York: Academic Press.

Halle, M. (1985). Speculations about the representation of words in memory. In V. A. Fromkin (ed.) (1985) *Phonetic linguistics: Essays in honor of Peter Ladefoged,* 101-14. London: Academic Press.

Harrikari, H. (1999). A Constraint-based approach to Finnish CV spoonerisms. *Nordic Journal of Linguistics, 22*(2), 111-135.

Harris, J. (1994). *English sound structure.* Blackwell, Oxford.

Jackendoff, R. (2002). *Foundations of language:* New York: Oxford University Press.

Johnson, E. P., Pennington, B, F., Lowenstein, J, H., & Nittrouer, S. (2011). Sensitivity to structure in the speech signal by children with speech sound disorder and reading disability. *Journal of Communication Disorders, 44,* 294–314.

Kaye, J. and Harris, J. (1990). Segmental complexity and phonological government. *Phonology,*7(1), 255–300.

Leahy, J & Dodd, B. (1987). The development of disordered phonology: A case study. *Language and Cognitive Processes, 2* (2), 115-132.

Locke, J., & Studdert-Kennedy, M. (1983). *Phonological acquisition and change.* New York: Academic Press.

McNeill, B. C., & Hesketh, A. (2010). Developmental complexity of the stimuli included in mispronunciation detection tasks. *International Journal of Language & Communication Disorders*, 45 (1), 72–82.

Ohala, J. (1983). The origin of sound patterns in vocal tract constraints. In MacNeilage, Peter F. (ed.), *The production of speech*. Heidelberg and Berlin: Springer Verlag.

Ohala, J. (1990). There is no interface between phonology and phonetics: A personal view. *Journal of Phonetics*, 18, 153-171.

Poeppel D. (2014). The neuroanatomic and neurophysiological infrastructure for speech and language. *Current Opinion in Neurobiology*, *28*, 142–149.

Reiss, C, & Volenic, V. (2017). Cognitive phonetics: The transduction of distinctive features at the phonology phonetics interface. *Biolinguistics 11,* 251-294.

Shriberg, L. D., & C. Widder. (1990). Speech and prosody characteristics of adults with mental retardation. *Journal of Speech and Hearing Research, 33*,627–653.

Shriberg, L. D., Fourakis, M., Karlsson, H. K., Lohmeier, H. L., McSweeney, J., Potter, N. L., et al. (2010). Extensions to the speech disorders classification system (SDCS). *Clinical Linguistics & Phonetics, 24*, 795–824.

Stackhouse, J., and Wells, B. (1997). *Children's speech and literacy difficulties I: A psycholinguistic Framework.* London: Whurr.

Sutherland, D. (2006). Phonological representations, phonological awareness, and print decoding ability in children with moderate to severe speech impairment. Doctoral Thesis. Christchurch, University of Canterbury.

Sutherland, D., & Gillon, G. (2005). Assessment of phonological representations in children with speech impairment. *Language Speech and Hearing Services in Schools*, *36*(4), 294-307.

Sutherland, D., & Gillon, G. T. (2007). The development of phonological representations and phonological awareness in children with speech impairment. *International Journal of Language and Communication Disorders, 42(2*), 229–250.

Terband, H., Maassen, B., and Maas, E. (2019). A psycholinguistic framework for diagnosis and treatment planning of developmental speech disorders. *Folia Phoniatrica Et Logopaedica, 71*(5-6), 216-227.

Waring, R., & Knight, R. (2013). How should children with speech sound disorders be classified? A review and critical evaluation of current classification systems. *International Journal of Language & Communication Disorders*, *48*(1), 25-40.

Characterizing Human vs Machine Gameplay in StarCraft II

Chad A. Peters (chad.peters@carleton.ca)
Institute of Cognitive Science, Carleton University
Ottawa, ON Canada

Babak Esfandiari (babak@sce.carleton.ca)
Department of Systems and Computer Engineering, Carleton University
Ottawa, ON Canada

Robert L. West (robert.west@carleton.ca)
Institute of Cognitive Science, Carleton University
Ottawa, ON Canada

Abstract

Our research presents a review of the StarCraft II ecosystem, and an analysis of those universal characteristics integral to the replay data generated by thousands of humans and robots in mixed competitions. In this paper we present the obvious and subtle differences between human and machine tournament play, and demonstrate that we can still identify and leverage various aspects of game play to distinguish human from machine.

Keywords: StarCraft; Real-Time Strategy; Behavior Modeling; Cognitive Modeling; Machine Learning; Artificial Intelligence

Introduction

In this paper we describe an analysis of Starcraft game playing replay data to draw attention to differences between human and machine gameplay style, and subtle indicators that may help an observer identify bots that otherwise play very much like a human, and are even capable of defeating expert human players in tournament play.

Our primary motive for replay analysis is less about finding out how to be the best human player, or how to design the best autonomous agent; rather, we are interested to see what sets humans apart from game-playing A.I., and how successfully these aspects of game play can be measured, modeled, and perhaps even used to detect, classify, and replicate these behaviors. We know that A.I. developers describe game playing bots in *terms* of human cognition, and challenge how much of what the bot is and does is actually useful for *understanding* human cognition.

In this paper we attempt to answer these questions through an investigation of game playing behavior produced by humans and machines, both individually, and when playing against each other. The rest of this paper is outlined as follows:

We first review the StarCraft game universe, with an overview of the most recent replay data format and analysis tools, common measurements and their significance, and how this information can be useful to inspect and understand game playing agent behavior in general.

Next, we discusses the human vs human competitive arenas, play styles and metadata that can be used to differentiate players of various skill levels. We find that all human players tend to trend towards a higher and varied rate of effective input as they get better.

The third section deals with machine vs machine tournament play, where A.I. developers can pit their agents against each other in an accelerated tournament environment to evaluate new techniques in intelligent agent design, and specifically for Real-Time Strategy environments such as Starcraft. Our analysis of machine agents demonstrates the tendency of bots to maximize all available action bandwidth provided by tournament servers, and make very little use of the features (or constraints) of a user interface.

In the last section, we discuss the results of recent competitive tournament play between the world's best humans and machines, their apparent similarities and subtle differences in behavior, and some of the controversy involved when A.I. tries to be only as human as necessary. Our results demonstrate that even the most human-like bots are still exploiting non-human abilities in competitions, and possibly disqualifies their use as a model of human cognition.

This paper concludes with a summary of human and robot play styles and indicators, the impact of recent events on the gaming community writ large, and possible future directions for research in this problem domain.

Overview of StarCraft II

The StarCraft II[1] game franchise is a space-opera set in a fictional universe featuring three intelligent racial factions vying for survival and control of limited resources as represented through a series of maps taking place across a variety of terrain. Each of the three playable races (or factions) made available to the player specialize in a unique style of warfare, with corresponding strengths and weaknesses (in Paper-Rock-Scissors fashion) that may appeal to different player preferences.

Real-Time Strategy (RTS) games such as StarCraft require players to successfully balance multiple elements such as resource management, dealing with uncertainty and imperfect information through fog of war, foresight and anticipation, and regularly switching between strategic macro- and tactical

[1] https://starcraft2.com/

micro-management (respectively) to optimize control of various units and groups. RTS games of this nature are known in gaming community and eSports circles for demanding a high level of cognitive performance from players and tend to draw out players with an acumen and appetite for thriving in complex, fast-paced, and high-stakes environments.

Like many RTS games of similar nature, SC2 provides a default set of key mappings that allow a player to quickly input key combinations to accomplish slightly more complex commands. Game command complexity ranges from simple single mouse clicks or keyboard shortcuts, to more complex keystroke combinations for unit group selection or navigating "tech trees" for producing and replacing various units. Command complexity can be measured through a combination of the number of keystrokes required to register a valid selection, and amount of time typically required to complete the input.

The StarCraft community uses different measurements to compare and contrast player performance and aptitude during a match, mostly focusing on the rate of input during different phases of gameplay; example of this include raw input and screen adjustments within a given period of time. The base measurement of Actions Per Minute (APM)[2] can be defined as the lowest level of user input typically associated with the push of a button on the keyboard or mouse. Observation of human replays demonstrates that a high APM, while frequently correlated with a high skill level, does not necessary predict effective gameplay, as many actions are simply repetitive clicking on the same area of the screen. This behavior is perhaps used by some players to maintain a certain micro-management tempo during escalated confrontation. High-frequency actions are not necessarily useful actions, and thus the literature sometimes makes use of Effective APM (EAPM or EPM), to distinguish strings of commands that are both unique and valid from those spammed in repetition.

Sc2gears[3], an online replay analysis site, makes an additional distinction between Micro- and Macro-APM: activities that require resources such as building, training, upgrading, or researching are considered macro-management activities and contribute to overall strategic play, whereas everything else involving individual units or groups for movement and direct engagement with the opponent are considered forms of micro-management.

Screens Per Minute (SPM), another common measurement describing manipulation of the visual playing field, can be defined in a similar way to APM; however, the low level operator in this case is the number of times the player moves the screen in one minute. Moving a screen in StarCraft can take the form of either panning (by using either keyboard or mouse to move the screen in one of four cardinal directions), or by selecting a specific spot on the mini-map.

[2] https://liquipedia.net/starcraft2/APM

[3] https://sites.google.com/site/sc2gears/features/replay-analyzer/apm-types

Figure 1: StarCraft II Game depicting a battle between Terran and Protoss forces

Related Work

Laird and Van Lent(2001) could be credited with one of the earliest initiatives to promote video games as an alternative platform for testing Artificial Intelligence. Different game genres attracted different audiences, from the idle Puzzle Adventure gamer to the hard-core (and somewhat twitchy) First Person Shooter (FPS); as recent history would have it, a combination of balanced and repeatable gameplay and backing from the eSports industry has projected the Real-Time Strategy (RTS) genre into the spotlight, attracting players from all walks of life – for fun, profit, and everything in between.

In (Robertson & Watson, 2014), RTS games like StarCraft have become a de-facto standard for training and testing learning agents, with a growing divide between academic research and the games industry. Researchers are discovering different ways of modeling and understanding spatial-temporal hierarchy problems, however, many papers use different evaluation metrics, making comparison extremely difficult.

Webber et al. (2010) studied players of various skill levels and found those with consistently higher APM usually perform better in RTS games such as StarCraft; their analysis concludes this is due to experts encoding ballistic action sequences. Further, the expert player produces a higher Spatial Variance of Action (distributed attention) yielding a higher probability of yielding required information without causing cognitive overload – professional players know where and what to look for without investing valuable focus time on arbitrary features (Weber et al., 2010).

In (Čertický & Churchill, 2017) we find a review of the current state of these competitions, and the variety of AI bots that compete in them. Growing interest from the gaming industry eventually led to a joint effort between Blizzard Entertainment[4] and Google Deepmind[5] in the creation of a publicly available StarCraft 2 Learning Environment to fast tract

[4] https://www.blizzard.com/

[5] https://deepmind.com/

development of Reinforcement Learning agents, and agents that learn to achieve a level of play that is comparable to a novice player (Vinyals et al., 2017).

Huang et al.(2017) found that some novice and most experts produce excessive APM during the first two minutes of a game; those interviewed associated it with a warm up (not unlike sports games); however, a distinguishing feature between novice and expert is the consistency of APM through the rest of the match – experts rarely decline in APM, whereas novices drop off during periods of intense or confusing states. Further, they also found that experts are more likely to use unit groups for production buildings (vs mobile units), rebind unit groups on-the-fly, and retain consistent habits regardless of game state.

Penney et al.(2018) explored the focus of attention using Information Foraging Theory, and found that while all players have to actively select what to focus on, experts have the highest "return of investment" for their efforts. At a macro-cognitive level, their participants favored *What* information over the *Why* as reported by previous research, and their *Whats* were nuanced, complex, and sometimes expensive, causing participants to dwell on features longer than necessary. They also found players' decision points fell into four main categories of decision cues: building/producing, fighting, moving, and scouting. They found player time spent processing these points were largely dominated by fighting and building, to the point that signs of fighting were classified as major distractor cues(Penney et al., 2018).

Methodology

Our analysis of player matches uses a combination of state-action pairs and metadata derived from StarCraft replay files. SC2 replays are stored in MoPaQ[6] (MPQ) files, an efficient container created by Blizzard Entertainment to store media and gameplay data. MPQ archives can store an arbitrary number of files to encapsulate game state and associated metadata for later retrieval, replay, and in this case, analysis of sequential state-action pairs.

There are various tools to parse the human- and machine-generated replay packs, discover those features most indicative of human players, and to model those features such that they can be compared against their machine-generated counterparts. We primarily used Blizzard's s2protocol[7], a Blizzard python library for decoding SC2Replay files, and Scelight[8], a replay visualization and report generator that excels at build order and ladder career analysis for competitive players.

Game event metadata, such as time stamp (consisting of game loops, each representing approximately 0.0625s), player ID, and command ID (Table 1). The SC2 command taxonomy consists of a relatively simple parent-child hierarchy with the most prevalent commands at the top. Table 1 gives and example of event IDs parsed from a replay file that can be used to categorize classes of user inputs.

ID#	Command
49	Camera Update
104	Cmd Update Target Point
29	Control Group Update

Table 1: Sample command types seen in replay data

All of these tools can be used to extract and visualize various aspects of game play such as current player league standing, command sequences, input (action) frequency, and game events that are presented to each player at specific times. We used this information to derive and compare the common measurements for each combination of human and machine match up as discussed in the rest of this paper.

Human vs Human

The Blizzard ladder league system ranks and matches Human players according to their evolving Match Making Ranking (MMR) score, which in turn is largely based on the Elo Chess Ranking system created by Arpad Elo and adopted by the US Chess Federation[9] to calculate relative skill level between players in organized competition. Blizzard games use Battle.net, a platform-independent system used to match and rank competitive players, and provide an API to access replay data archives associated with each match. The MMR scoring system used by Blizzard is also used to divide players into leagues for general comparison and occasional tournament organization.

Data Sources

In late 2017, Blizzard and Deepmind embarked on a joint effort to create and release a set of tools that could be used to accelerate research and development of intelligent agents in this domain, including a corpus[10] of anonymized human-generated replays for use by researchers wishing to model human players, and for training and testing associate Reinforcement Learning algorithms. Blizzard then released the replays in two sets; the first set is a stratified sample of 64,396 matches, and is a subset of a more complete historical corpus consisting of 1,160,650 replays. Our analysis of human performance characteristics made use of the smaller of the two sets to as it provided sufficient representation of all skill levels (as represented by player Match Making Ranking and placement league), and still feasible for most researchers to reproduce on standard computational resources.

Upon closer inspection of the replay files, we found that each header, although anonymized by player ID, still contained the player's per-match MMR in the replay header metadata. We extracted the MMR along with the average APM per player for each of the 64k replay files to establish a

[6] https://fileinfo.com/extension/mpq
[7] https://github.com/Blizzard/s2protocol
[8] https://sites.google.com/site/scelight/
[9] http://www.uschess.org/about/about.php
[10] https://github.com/Blizzard/s2client-proto#replay-packs

more accurate correlation between observable characteristics and MMR. We then filtered replays due to missing or corrupted headers, incomplete or unknown matches, and missing League information. We resumed with refined corpus of 45,834 replays, each representing two different players, for a maximum of 91,668 unique *plays*.

Replay Analysis

Our overview begins with features that can be easily aggregated, namely APM, SPM, and MRR. The averages across all replays was 90 APM, and 8.42 SPM (Table2).

Player League	Plays	Plays %	Avg APM	Avg SPM
Grandmaster	2,447	2.67%	193	21.64
Master	9,700	10.58%	155	15.68
Diamond	34,526	37.66%	105	9.50
Platinum	18,395	20.07%	76	6.77
Gold	13,308	14.52%	59	5.49
Silver	10,947	11.94%	47	4.52
Bronze	2,316	2.53%	39	3.75

Table 2: Player APM and SPM distribution by League

In Figure 2 we see a moderately positive correlation ($r = 0.65$) between player APM and MMR extracted from each SC2 replay file metadata. This indicates a relationship between the two, with higher average APM likely being the result of player skill level, rather than the cause of it. Breaking out player APM by League standing (Figure3), we observe an increase in both mean and variance of player APM as player skill rises through beginner to expert levels.

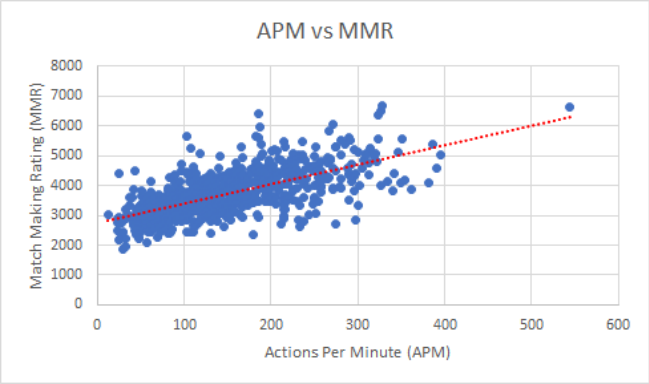


Figure 2: Player APM by MMR; $r = 0.65$

The increase in mean APM is somewhat expected, as players familiar with the game and action sequences will naturally initiate, queue up, and respond to in-game actions on the fly. We also find the increase in APM variance an interesting phenomenon, which could indicate a divergence of macro (strategic) vs micro (tactical) play styles; however, this could also be due to a higher upper bound limited by each player. Each league is normally distributed around the mean with a long tail in the upper-APM range.

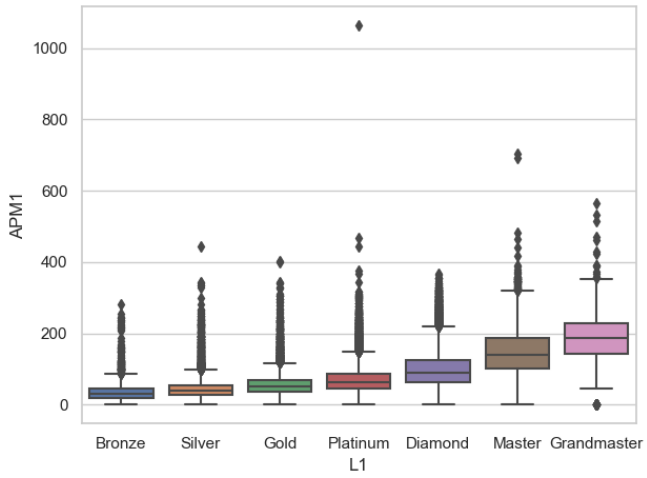


Figure 3: PlayerAPM Quartile Range per League

If we aggregate the average APM and SPM by player league, there is indeed noticeable relationship between league and average APM. We also identified two extreme outliers with one Platinum league player with an average APM of 1064, and two Master league players with 704 and 691 average APM. In both of the later two games we observed cycling between static building control groups multiple times per second. A detailed breakdown of this replay revealed APM spikes as high as 1479 from cycling through control groups; behavior performed at millisecond intervals is only typical of bots observed in the AI arenas, and is considered against the EULA for ranked human vs human Ladder matches.

Machine vs Machine

Bot vs bot gameplay is a popular method to test the efficiency of machine agents in RTS games. We next review are few of the most currently active AI tournament managers that cater to AI research for Starcraft 2.

Data Sources

The SC2 AI Community[11] is one of the most active and longstanding in machine vs machine competitive play, provides ample resources to introduce developers to create agents based on working examples, and also hosts an ongoing Ladder-style matchup service for competitive bot vs bot matches. The StarCraft 2 AI Ladder system continually ranks bots using a similar Elo-based calculation, and can be used as a rough indicator of skill for theoretical league divisions; this ranking system can be used as a rough guide, however we found no evidence of SC2AI bots being divided into leagues as Blizzard does with human players.

[11] https://sc2ai.net/

We first obtained samples of the last few weeks of Season 8 ladder ranking in machine vs machine matches to set a baseline of activity. To do this, we downloaded a minimum of 20 matches per active bot, subject to posted replay availability and game completion, for a total 846 replays. After filtering for crashed games and replay errors, we obtained 798 verified replays. Of these bots, only 10 have made their code available for public inspection, whereas the rest have reserved their right to keep the code private, and only disclose the compiled DLL or Bytecode for offline play. The ability to reserve the ability to choose public or private source code is a feature that allows competitive researchers to participate with a lower risk of losing intellectual property.

Replay Analysis

We found the indicators of interest over all replays differed substantially from our human-only replay results. As seen in Figure 4, the Average APM for all machines was 3465; a roughly 10-fold increase that, while somewhat expected of autonomous agents in this environment, confirms that this machine-only environment is not limited to the same timing constraints as their human counterparts, and will use it when available.

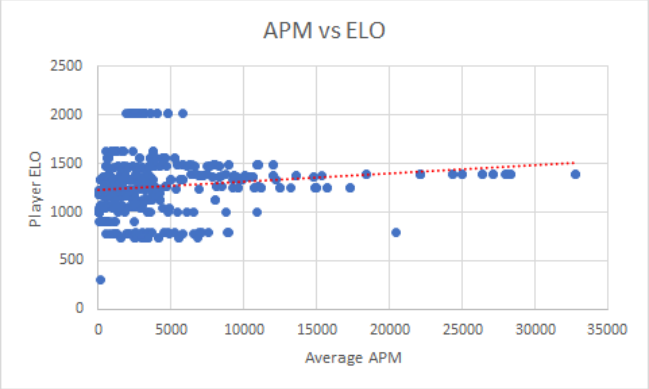


Figure 4: Bots APM by ELO rating; $r = 0.15$

Length range	Plays	Plays %	Avg APM	Avg SPM
0-5 min	47	4.92%	1,830	0.00
5-10 min	426	44.56%	3,926	0.00
10-15 min	346	36.19%	3,726	0.01
15-20 min	71	7.43%	3,306	0.00
20-25 min	24	2.51%	2,208	0.00
25-30 min	4	0.42%	1,641	0.00
35-40 min	8	0.84%	1,431	0.00
45-50 min	30	3.14%	1,025	0.00

Table 3: Bot games by match length: matches have a higher APM by time, negligible use of screen resources.

Also of interest is the almost non-existent use of the screen interface, as reflected by null values for Avg SPM (Table 3). The authors agree that this is likely due to the prolific use of low-level APIs made available to agent developers that allow for access to all permissible game state information regardless of actual location – in essence giving every agent a birds-eye view of the entire match.

Only 5 sessions register Screens Per Minute (SPM), ranging from 0.05 (every 3 seconds) to 0.24 (every 14 seconds). Upon closer inspection there were 4 bots issuing valid Screen Update commands, however, additional investigation is required to determine the causal relationship between observed state and action sequencing without running a code trace.

Correlation coefficient between Elo score and APM is only slightly positive (0.15), indicating the lack of an upper bound on event generation is not a competitive advantage for machines; rather the sophistication of processing (or lack thereof) is more at play. The per-match APM for the majority of bots are between 755 and 3775 with an IQR of 3030, with upper bound outliers going as high as 35,000 APM. This not only captures the majority of games, but the lower bound is well above the peak threshold of professional human players (600), and provides a clear threshold for classification.

Human vs Machine

In the last section, we discuss the results of various tournament play between the world's best humans and machines, derived from a variety of sources. We found that access to quality replays of this nature is somewhat more difficult to obtain, but comparable across game versions using standard metrics as above.

Data Sources

Our first dataset was obtained from the Aritificial Intelligence Starcraft Tournament (AIST) platform[12], that operates tournaments in a hybrid-like fashion. AIST regulations are similar to the well known Student StarCraft AI Tournament (SSCAIT)[13]; however AIST is unique in that they invite high ranking SC:BW players to compete against each seasons' winning bot. We downloaded replays of the final human vs bot matches from the last 3 seasons for a total of 18 replays; all replays were imported without issue.

Our second dataset consisted of a mixture of formal and informal matches played by the well-known AlphaStar[14] agent designed by Google DeepMind. Replay sources consisted of ten (10) public tournament rounds against two WCS champions (TLO and MaNa). The second source consisted of a mixture of replays representing the progression of three learning phases against human players on Battle.net, as discussed by Vinyals et al.(2019).

[12] https://sites.google.com/view/aistarcrafttournament/

[13] https://sscaitournament.com/

[14] https://deepmind.com/research/open-source/alphastar-resources

Replay Analysis

The AIST replays demonstrate a common theme between populations of humans and machines, with a higher level APM vs MMR/Elo of bots. Figure5 contains the aggregate of all tournament rounds between human (H) and machine (M) for the last two seasons, with APM distributions clearly demarcating between the two. AlphaStar, though still quite a bit faster than humans through sustained bursts of high APM, does not make it as obvious through clever use how APM is calculated.

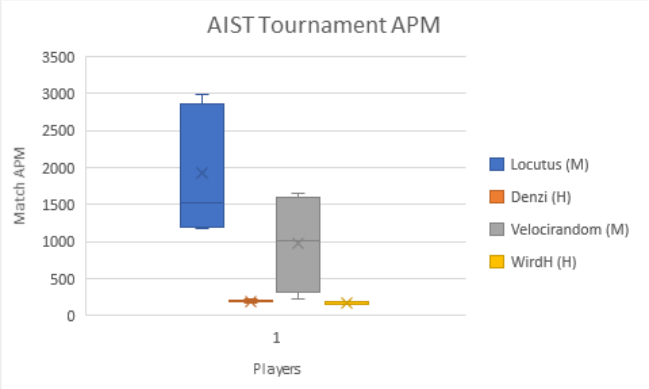


Figure 5: AIST APM for HvM matchup: the two machines have a much higher APM maximum and IQR than their human counterparts.

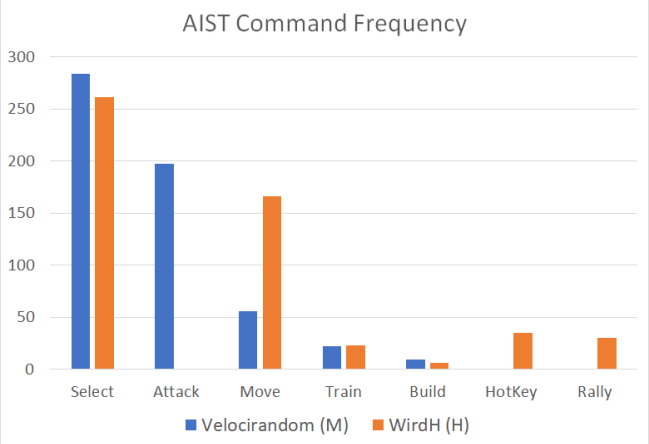


Figure 6: AIST command frequency: humans make greater use of complex commands and unit groupings.

Both replay sources indicate the bots are making direct use of the API, without using a mouse, keyboard, or screen. The ability to see the entire game at once is something of a grey area (but was not against the rules at the time); however, it is still an advantage reserved for agents since human players can only see a small slice of the map at a time. In (Vinyals et al., 2019) AlphaStar tends to focus its attention to one area at a time, which is *sort of* like a human player moving their camera around; however, humans are still at a disadvantage as they still have to move a singular point of focus while using camera controls, at the same time. (Lin et al., 2019) corroborated this with early supervised and mid-tier replays.

With these similarities in mind we took another look at how each of the players interacted with the game itself, and found that all bots, regardless of tournament or game version, do not make use Unit Group Hotkeys (in or out of combat, Figure6). This may appear surprising, however, does make sense as it adds a layer of complexity on top existing data structures, and is only detectable after-the-fact during replay analysis.

Conclusion

In this paper we presented an analysis of StarCraft replay data generated by a variety of sources, to characterize distinguishing features between humans and machines. We provided an overview of the StarCraft tournament ecosystem, and its potentials for both A.I. development and understanding of human cognition.

Our Human vs Human results set the standard for typical performance in raw Actions Per Minute, with is a positive correlation between player APM and skill level in humans, and a few artifacts suggesting some players are trying to use bots or auto-scripting to game the ladder system. This behavior is in stark contrast to the replays evidenced by Machine vs Machine tournaments, where there is a weak relationship between action speed and ranking; agent architectures will take advantage of as much computing resource as possible without consideration for human-likeness or limitations, if they don't have to. Last, we observed a mixed adherence to human-likeness in Human vs Machine match-ups. Despite the shrinking gap between obvious and subtle play styles, in-depth analysis shows us that even the most sophisticated bots will not make use of interface features designed for humans, if they don't absolutely have to.

While some of the results of this research are compelling, there are areas that require additional investigation. We have begun to model game playing events in terms of discrete and continuous stochastic processes through Markov transitions, and would like to continue investigating probable internal representations of behavior using Hidden Markov Models. We hope these and other areas of investigation will shed additional light on what distinguishes a human from a machine while still engaging on common ground.

References

Čertický, M., & Churchill, D. (2017). The Current State of StarCraft AI Competitions and Bots. In *Proceedings of the aiide 2017 workshop on artificial intelligence for strategy games.*

Huang, J., Yan, E., Cheung, G., Nagappan, N., & Zimmermann, T. (2017). Master Maker: Understanding Gaming Skill Through Practice and Habit From Gameplay Behavior. *Topics in Cognitive Science*, *9*(2), 437–466. doi: 10.1111/tops.12251

Laird, J. E., & Van Lent, M. (2001). Human-level AI's killer application interactive computer games. *AI Magazine*, *22*(2), 15–25.

Lin, M., Wang, T., Li, X.-B., Liu, J., Wang, Y., Zhu, Y., & Wang, W.-P. (2019). An Uncertainty-Incorporated Approach to Predict the Winner in StarCraft II Using Neural Processes. *IEEE Access*, *7*, 101609–101619. doi: 10.1109/access.2019.2930581

Penney, S., Dodge, J., Hilderbrand, C., Anderson, A., Simpson, L., & Burnett, M. (2018). Toward foraging for understanding of StarCraft agents: An empirical study. *International Conference on Intelligent User Interfaces, Proceedings IUI*, 225–237. doi: 10.1145/3172944.3172946

Robertson, G., & Watson, I. (2014). A Review of Real-Time Strategy Game AI. *AI Magazine*, *35*(4), 75. Retrieved from https://aaai.org/ojs/index.php/aimagazine/article/view/2478 doi: 10.1609/aimag.v35i4.2478

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Georgiev, P., Oh, J., ... Apps, C. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*, 350–354. doi: 10.1038/s41586-019-1724-z

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., ... Tsing, R. (2017). *StarCraft II: A New Challenge for Reinforcement Learning.* Ithaca, New York: arXiv:1708.04782. Retrieved from http://arxiv.org/abs/1708.04782

Weber, B. G., Mateas, M., & Jhala, A. (2010). Applying Goal-Driven Autonomy to StarCraft. In *Proceedings of the sixth aaai conference on artificial intelligence and interactive digital entertainment* (pp. 101–106). Retrieved from http://aaai.org/ocs/index.php/AIIDE/AIIDE10/paper/view/2142

Amortized Bayesian Inference for Models of Cognition

Stefan T. Radev
Heidelberg University, Germany

Andreas Voss
Heidelberg University, Germany

Eva Marie Wieschen
Heidelberg University, Germany

Paul-Christian Bürkner
Aalto University, Finland

Abstract

As models of cognition grow in complexity and number of parameters, Bayesian inference with standard methods can become intractable, especially when the data-generating model is of unknown analytic form. Recent advances in simulation-based inference using specialized neural network architectures circumvent many previous problems of approximate Bayesian computation. Moreover, due to the properties of these special neural network estimators, the effort of training the networks via simulations amortizes over subsequent evaluations which can re-use the same network for multiple datasets and across multiple researchers. However, these methods have been largely underutilized in cognitive science and psychology so far, even though they are well suited for tackling a wide variety of modeling problems. With this work, we provide a general introduction to amortized Bayesian parameter estimation and model comparison and demonstrate the applicability of the proposed methods on a well-known class of intractable response-time models.

Keywords: Bayesian inference; Neural networks; Cognitive models; Deep learning; Simulation

Generative Models in Cognitive Science

Mathematical models formalize theories of cognition and enable the systematic investigation of cognitive processes through simulations and testable predictions. They enable a systematic joint analysis of behavioral and neural data, bridging a crucial gap between cognitive science and neuroscience (B. M. Turner, Forstmann, Steyvers, et al., 2019). Moreover, questions demanding a choice among competing cognitive theories can be resolved at the level of formal model comparison.

The *generative* property of such models arises from the fact that one can simulate the process of interest and study how it behaves under various conditions. More formally, consider a cognitive model $\mathcal{M}$ which represents a theoretically plausible, potentially noisy, process by which observable behavior x arises from an assumed cognitive system governed by hidden parameters θ and an independent source of noise $\xi \sim p(\xi)$:

$$x = \mathcal{M}(\theta, \xi) \tag{1}$$

Generative models of this form have been developed in various domains throughout psychology and cognitive science, including decision making (Voss, Lerche, Mertens, & Voss, 2019), memory (Myung, Montenegro, & Pitt, 2007), reinforcement learning (Fontanesi, Gluth, Spektor, & Rieskamp, 2019), risky behavior (Stout, Busemeyer, Lin, Grant, & Bonson, 2004), to name just a few. Once a model (or a set of models) of some cognitive process of interest has been formulated, the challenge becomes to perform inference on real data. We will now briefly review the mathematical tools provided by Bayesian probability theory for parameter estimation and model comparison (Jaynes, 2003). Then, we will peruse a novel framework for performing Bayesian inference on models of cognition which are intractable with standard Bayesian methods.

Bayesian Parameter Estimation

Bayesian parameter estimation leverages prior knowledge about reasonable parameter ranges and integrates this information with the information provided by the data to arrive at a *posterior distribution* over parameters. In a Bayesian context, the posterior encodes our updated belief about plausible parameter ranges conditional on a set of N observations $X := \{x_n\}_{n=1}^{N}$. Bayes' rule gives us the well known analytical form of the posterior:

$$p(\theta \mid X) = \frac{p(X \mid \theta)\, p(\theta)}{\int p(X \mid \theta)\, p(\theta)\, d\theta} \tag{2}$$

where $p(X \mid \theta)$ represents the *likelihood* of the parameters θ and $p(\theta)$ denotes the *prior*, that is the distribution of θ before observing the data. The denominator is a normalizing constant usually referred to as the *marginal likelihood* or *evidence*. Note, that all distributions are also implicitly conditional on the particular generative model $\mathcal{M}$.

Based on the obtained estimate of the posterior distribution, usually in the form of random draws from the posterior, summary statistics such as posterior means or credible intervals for each parameter can be obtained. What is more, the posterior distribution can be further transformed to obtain subsequent quantities of interest, for example, the *posterior predictive distribution* which can be compared to the observed data for the purpose of model checking (Lynch & Western, 2004).

Bayesian Model Comparison

In many research domains, there is not a *single model* for a particular process, but whole *classes* of models instantiating different and often competing theories. Bayesian model comparison proceeds by assigning a plausibility value to each candidate model. These plausibility values (model weights,

model probabilities, model predictions, etc.) can be used to guide subsequent model selection.

To set the stage, consider a set of J candidate models $\mathcal{G} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_J\}$. An intuitive way to quantify plausibility is to consider the marginal likelihood of a model $\mathcal{M}$ given by:

$$p(X \mid \mathcal{M}) = \int p(X \mid \theta, \mathcal{M})\, p(\theta \mid \mathcal{M})\, d\theta \tag{3}$$

which is also the denominator in Eq.2 (with $\mathcal{M}$ implicit in the previous definition). This quantity is also known as *evidence*, or *prior predictive* distribution, since the likelihood is weighted by the prior (in contrast to a posterior predictive distribution where the likelihood would be weighted by the posterior). The marginal likelihood penalizes the prior complexity of a model and thus naturally embodies the principle of Occam's razor (Jaynes, 2003). To compare two competing models, one can focus on the ratio between two marginal likelihoods, called a Bayes factor (BF):

$$\mathrm{BF}_{ij} = \frac{p(X \mid \mathcal{M}_i)}{p(X \mid \mathcal{M}_j)} \tag{4}$$

which quantifies the relative evidence of model i over model j. Alternatively, if prior information about model plausibility is available, one can consider model posteriors $p(\mathcal{M} \mid X) \propto p(X \mid \mathcal{M})\, p(\mathcal{M})$ and compute the posterior odds:

$$\frac{p(\mathcal{M}_i \mid X)}{p(\mathcal{M}_j \mid X)} = \frac{p(X \mid \mathcal{M}_i)}{p(X \mid \mathcal{M}_j)} \frac{p(\mathcal{M}_i)}{p(\mathcal{M}_j)} \tag{5}$$

which combine the relative evidence given by the BF with prior information in the form of prior odds.

Model Intractability

In order for cognitive models to be useful in practice, parameter estimation and model comparison should be feasible within reasonable time limits. As evident from their definitions, both Bayesian parameter estimation and model comparison depend on the likelihood function $p(X \mid \theta, \mathcal{M})$ which needs to be evaluated analytically or numerically for any triplet $(\mathcal{M}, \theta, X)$.

When this is possible, standard Bayesian approaches for obtaining random draws from the posterior, such as Markov chain Monte Carlo (MCMC), or optimizing an approximate posterior, such as variational inference (VI), can be readily applied. However, when the likelihood function is not available in closed-form or too expensive to evaluate, standard methods no longer apply.

In fact, many interesting models from a variety of domains in cognitive science and psychology turn out to be intractable (Voss et al., 2019; B. Turner, Sederberg, & McClelland, 2016). This has precluded the wide exploration and application of these models, as researchers have often traded off complexity or neurocognitive plausibility for simplicity in order to make these models tractable. In the following, we discuss the most popular approach to inference with intractable models.

Simulation-Based Inference

Simulation-based methods leverage the generative property of mathematical models by treating a particular model as a *scientific simulator* from which synthetic data can be obtained given any configuration of the parameters. Simulation-based inference is common to many domains in science in general (Cranmer, Brehmer, & Louppe, 2019) and a variety of different approaches exist. These methods have also been dubbed *likelihood-free*, which is somewhat unfortunate, since the likelihood is implicitly defined by the generative process and sampling from the likelihood is realized through the stochastic simulator:

$$x_n \sim p(x \mid \theta, \mathcal{M}) \iff x_n = \mathcal{M}(\theta, \xi_n) \text{ with } \xi_n \sim p(\xi) \tag{6}$$

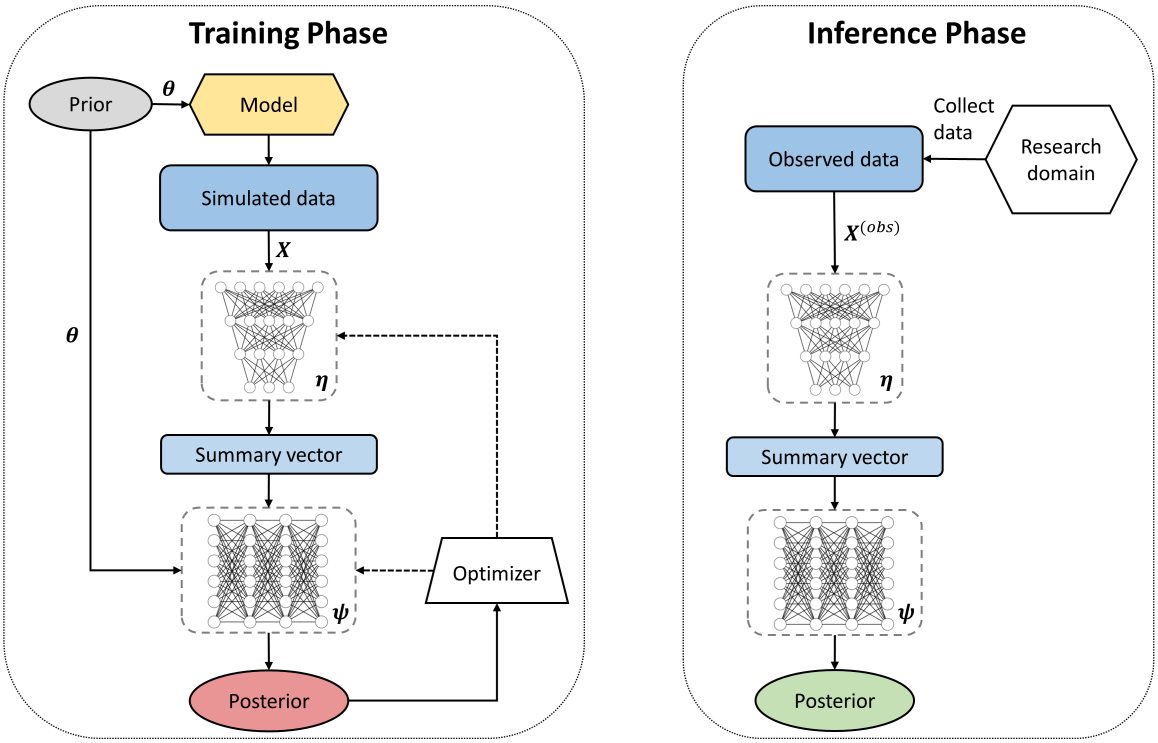
Different simulation-based methods differ mainly with respect to how they utilize the synthetic data to perform inference on real observed data (Cranmer et al., 2019). The utility of any simulation-based method depends on multiple factors, such as asymptotic guarantees, data utilization, efficiency, scalability, and software availability.

Approximate Bayesian computation (ABC) offers a standard set of theoretically sound methods for performing inference on intractable models (Cranmer et al., 2019). The core idea of ABC methods is to approximate the posterior by repeatedly sampling parameters from a proposal (prior) distribution and then generating a synthetic dataset by running the simulator with the sampled parameters. If the simulated dataset is sufficiently similar to an actually observed dataset, the corresponding parameters are retained as a sample from the desired posterior, otherwise rejected. However, in practice, ABC methods are notoriously inefficient and suffer from various problems, such as the *curse of dimensionality* or *curse of inefficiency* (Marin, Pudlo, Estoup, & Robert, 2018). More efficient methods employ various techniques to optimize sampling or correct potential biases.

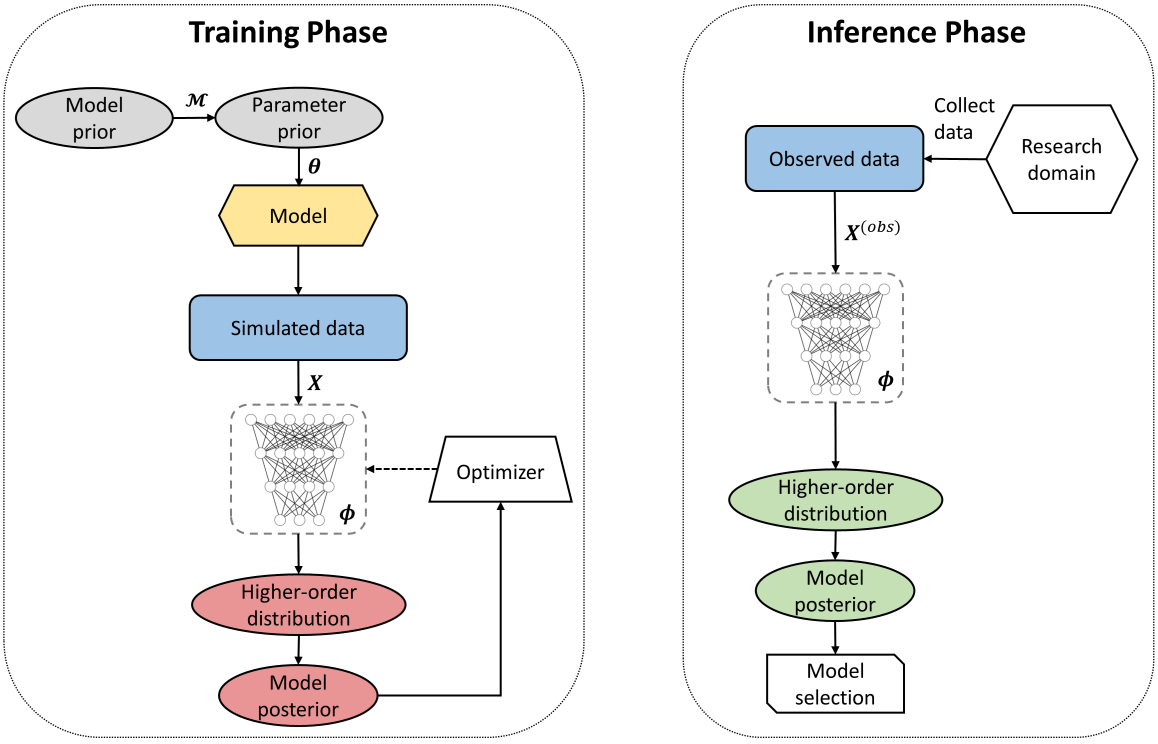
Recently, the scientific repertoire for simulation-based inference has been enhanced with ideas from deep learning and neural density estimation (NDE) in particular (Greenberg, Nonnenmacher, & Macke, 2019). These methods employ specialized neural network architectures which are trained with simulated data to perform efficient and accurate inference on previously intractable problems (Cranmer et al., 2019). NDE methods are rapidly developing and still largely underutilized in cognitive modeling, even though first applications to simulated (Radev, Mertens, Voss, Ardizzone, & Köthe, 2020; Radev, D'Alessandro, et al., 2020) as well as actual data (Wieschen, Voss, & Radev, 2020) exist.

Amortized Inference

The majority of simulation-based methods need to be applied to each dataset separately. This quickly becomes infeasible when multiple datasets are to be analysed and multiple candidate models are considered, since the expensive inference procedure needs to be repeated from scratch for each combination of dataset and model.



(a) Amortized parameter estimation



(b) Amortized model comparison

Figure 1: Graphical illustration of amortized parameter estimation and model comparison with different neural network estimators. (**a**) Amortized Bayesian parameter estimation with invertible neural networks (Radev, Mertens, et al., 2020). The left panel depicts the training phase in which the summary (f_η) and the inference network (f_ψ) are jointly optimized to approximate the true target posterior. The right panel depicts inference with already trained networks on observed data; (**b**) Amortized Bayesian model comparison with evidential neural networks (Radev, D'Alessandro, et al., 2020). The left panel depicts the training phase during which the evidential network f_ϕ is optimized to approximate the true model posteriors via a higher-order Dirichlet distribution. The right panel depicts inference with an already trained evidential network; the upfront training effort for both inference tasks is amortized over arbitrary numbers of datasets from a research domain.

In contrast, the concept of *amortized inference* refers to an approach which minimizes the cost of inference by separating the process into an expensive training (optimization) phase and a cheap inference phase which can be easily repeated for multiple datasets or models without computational overhead. Thus, the effort of training or optimization amortizes over repeated applications on multiple datasets or models. In some cases, the efficiency advantage of amortized inference becomes noticeable even for a few datasets (Radev, Mertens, et al., 2020; Radev, D'Alessandro, et al., 2020).

The field of amortized inference is rapidly growing and a variety of methods and concepts are currently being explored. For instance, *inference compilation* involves pre-training a neural network with simulations from a generative model and then using the network in combination with a probabilistic program to optimize sampling from the posterior (Le, Baydin, & Wood, 2016). The *pre-paid* estimation method (Mestdagh, Verdonck, Meers, Loossens, & Tuerlinckx, 2019) proceeds by creating a large grid of simulations which are reduced to summary statistics and stored on disk. Subsequent inference involves computing the nearest neighbors of an observed dataset in the pre-paid grid and interpolation. Sequential neural posterior estimation (SNPE) methods employ various iterative refinement schemes to transform a proposal distribution into the correct target posterior via expressive NDEs trained over multiple simulation rounds (Greenberg et al., 2019).

In line with these ideas, we recently proposed two general frameworks for amortized Bayesian parameter estimation and model comparison based on specialized neural network architectures (Radev, Mertens, et al., 2020; Radev, D'Alessandro, et al., 2020). In particular, these frameworks were designed to implement the following desirable properties:

- Fully amortized Bayesian inference for parameter estimation and model comparison of intractable models
- Asymptotic theoretical guarantees for sampling from the true parameter and model posteriors
- Learning maximally informative summary statistics directly from data instead of manual selection
- Scalability to high-dimensional problems through considerations regarding the probabilistic symmetry of the data
- Implicit preference for simpler models based purely on generative performance
- Online learning eliminating the need for storing large grids or reference tables
- Parallel computations and GPU acceleration applicable to both simulations, training/optimization, and inference

In the following, we describe our recently developed methods parameter estimation and model comparison in turn.

Amortized Parameter Estimation with Invertible Neural Networks

Recently, we proposed a novel amortization method based on invertible neural networks (Radev, Mertens, et al., 2020), which we dubbed *BayesFlow*. The method relies solely on simulations from a process model in order to learn and calibrate the full posterior over all possible parameter values and observed data patterns.

The BayesFlow method involves two separate neural networks trained jointly. A permutation invariant *summary network* is responsible for reducing an entire dataset X with a variable number N of *i.i.d.* observations[1] into a vector of *learned summary statistics*. Importantly, permutation invariant networks can deal with *i.i.d.* sequences of variable size and preserve their probabilistic symmetry. An *inference network*, implemented as an invertible neural network (Radev, Mertens, et al., 2020), is responsible for approximating the true posterior of model parameters given the output of the summary network. Invertible networks can perform asymptotically exact inference and scale well from simple low-dimensional problems to high-dimensional distributions with complex dependencies. During training, model parameters and synthetic datasets are generated on the fly and neural network parameters are adjusted via joint backpropagation (see Figure 1a, left panel, for a graphical illustration of the training phase).

Given a model and a prior over the model parameters, the goal is thus to train a conditional invertible neural network f_ψ with adjustable parameters ψ together with a summary network f_η with adjustable parameters η. These networks jointly learn an approximate posterior $p_\psi(\theta \,|\, f_\eta(X))$ over the relevant parameters for arbitrary numbers of datasets and dataset sizes N, as long as they share the same data structure. To achieve this, the networks minimize the Kullback-Leibler (KL) divergence between the true and the approximate posterior:

$$\min_{\psi,\eta} \mathbb{KL}\left(p(\theta \,|\, X) \,||\, p_\psi(\theta \,|\, f_\eta(X))\right) \tag{7}$$

Utilizing the fact that we have access to the joint distribution $p(\theta, X) = p(\theta)\,(X \,|\, \theta)$ via the simulator, we minimize the KL divergence in expectation over all possible datasets that can be generated given the prior and the model, resulting in the following optimization criterion:

$$\min_{\psi,\eta} \mathbb{E}_{p(\theta,x)}\left[-\log p_\psi(\theta \,|\, f_\eta(X))\right] \tag{8}$$

In practice, we approximate the criterion via its Monte Carlo (MC) estimate, since we can simulate theoretically infinite amounts of data and can easily evaluate $p_\psi(\theta \,|\, f_\eta(X))$ due to our invertible architecture. In case of perfect convergence of the networks, the summary network outputs *sufficient summary statistics* and the inference network samples from the true posterior (Radev, Mertens, et al., 2020). Importantly,

[1] Note, that the *i.i.d.* assumption is not a necessary condition for the method to work, but used here only to simplify the discussion.

once the networks have been trained with sufficient amounts of simulated data, they can be stored and applied for inference on multiple datasets from a research domain (see Figure 1a, right panel).

Amortized Model Comparison with Evidential Neural Networks

In another recent work (Radev, D'Alessandro, et al., 2020), we explored a framework for Bayesian model comparison on intractable models via *evidential neural networks*. We proposed to train a permutation invariant classifier network on simulated data from multiple models. The goal of this network is to approximate posterior model probabilities as accurately as possible. To achieve this, the network is trained to output the parameters of a higher-order probability distribution (parameterized as a Dirichlet distribution) over the model probabilities themselves, which quantifies the uncertainty in model probability estimates. Thus, for a classifier network with parameters ϕ, the higher-order posterior distribution over model probabilities is given by:

$$\text{Dir}(\pi \,|\, \alpha_\phi(X)) = \frac{1}{B(\alpha_\phi(X))} \prod_{j=1}^{J} \pi^{\alpha_\phi(X)_j - 1} \tag{9}$$

where $\alpha_\phi(X)$ denotes the vector of *concentration parameters* obtained by the network for a dataset X and $B(\cdot)$ is the multivariate *beta* function. The mean of this Dirichlet distribution can be used as a best estimate for the posterior model probabilities:

$$p_\phi(\mathcal{M} \,|\, X) = \frac{\alpha_\phi(X)}{\sum_{j=1}^{J} \alpha_\phi(X)_j} \tag{10}$$

Additionally, its variance can be interpreted as the epistemic uncertainty surrounding the actual evidence which the data provide for model comparison.

For training the network, we again utilize the fact that we have access to the joint distribution $p(\mathcal{M}, \theta, X)$ via simulations (see Figure 1b, left panel). Our optimization criterion is:

$$\min_\phi \mathbb{E}_{p(\mathcal{M},\theta,X)} \left[\mathcal{L}\left(p_\phi(\mathcal{M} \,|\, X), \mathcal{M}\right) \right] \tag{11}$$

where $\mathcal{L}(\cdot,\cdot)$ is a *strictly proper* loss function (Gneiting & Raftery, 2007), $\mathcal{M}$ is the true model index and the data X implicitly depend on θ. In practice, we approximate this expectation via draws from the joint distribution available via the simulator. Optimization of a strictly proper criterion, asymptotic convergence implies that the mean of the Dirichlet distribution represents the true model posteriors. Moreover, our simulation-based approach implicitly captures a preference for simpler models (Occam's razor), since simpler models will tend to generate more similar datasets. As a consequence, when such datasets are plausible under multiple models, the comparably simpler models will be more probable.

As with parameter estimation, once the evidence network has been trained on simulated data from the candidate models, it can be applied to multiple upcoming observations from a research domain (see Figure 1b, right panel).

Example Applications

In the following, we will present two applications of amortized Bayesian parameter estimation to a recently proposed and intractable evidence accumulation model (EAM). The first illustrative application is a simulation study aimed at quantifying parameter recovery as a function of data set size. Such simulations are especially useful for planing experiments but usually too costly to perform in complex modeling scenarios. The second application is concerned with parameter estimation on real data and serves as an illustration on how researchers might utilize amortized Bayesian inference with a pre-trained density estimator in practice.

EAMs are a popular class of models in psychology and cognitive science, as they allow a model-based analysis of response time (RT) distributions. Here, we will consider a Lévy flight model (LFM) with a non-Gaussian noise assumption (Voss et al., 2019; Wieschen et al., 2020) as an example. The Lévy flight process is driven by the following stochastic ordinary differential equation (ODE):

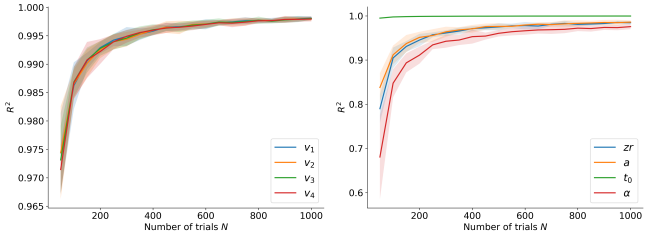
$$dx_c = v_c\, dt + \xi dt^{1/\alpha} \tag{12}$$

$$\xi \sim \text{AlphaStable}(\alpha, 0, 1, 0) \tag{13}$$

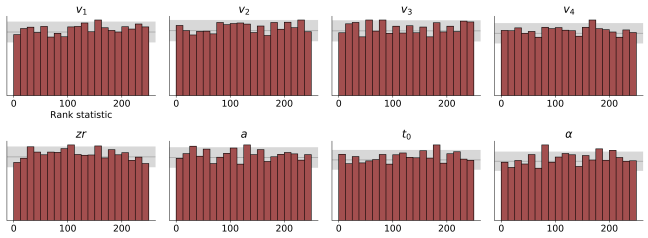
where dx_c denotes accumulated cognitive evidence in condition c, v_c denotes the average speed of information accumulation (drift), and α controls how heavy the tails of the noise distribution are (i.e., smaller values increase the probability of outliers in the accumulation process). Further parameters of the model are: a decision threshold (a) which reflects the amount of information needed for selecting a response; a starting point (z_r) indicative of response biases; and a non-decision time (t_0) reflecting additive encoding and motor process. Since the relationship of the α parameter to the standard parameters of the classical diffusion model (Ratcliff, Thapar, Gomez, & McKoon, 2004) has not been previously investigated, we focus on quantifying posterior correlations in the real data application.

Simulation Example

As a first example, consider a simulated RT experiment with four conditions. How many trials are needed for accurate parameter recovery? To answer this question, we can simulate multiple experiments with varying number of trials per participant (N) and then compute some discrepancy between ground-truth parameters and their estimates. However, since the model is intractable, such a simulation scenario is not feasible with non-amortized methods, which would need weeks on standard machines (Voss et al., 2019). However, using the *BayesFlow* method (Figure 1a), we can train the networks with simulated datasets and vary the number of trials during each simulation. Such a training takes approximately one day on a standard laptop equipped with an NVIDIA® GTX1060 graphics card. Subsequent inference is then very cheap, as amortized parameter estimation on 500 simulated participants takes less than 2 seconds.



(a) Parameter recovery as a function of trial numbers (N)



(b) Simulation-based calibration

Figure 2: Simulation results. (**a**) The left panel depicts parameter recovery of the four drift rate parameters as a function of trial numbers per participant N. The right panel depicts recovery of the other four parameters. Posterior means are used as summaries of the full posteriors and shaded regions represent bootstrap 95% confidence intervals. (**b**) The panel depicts simulation-based calibration (SBC) results at $N = 800$ as a validation check for the correctness of the full posteriors.

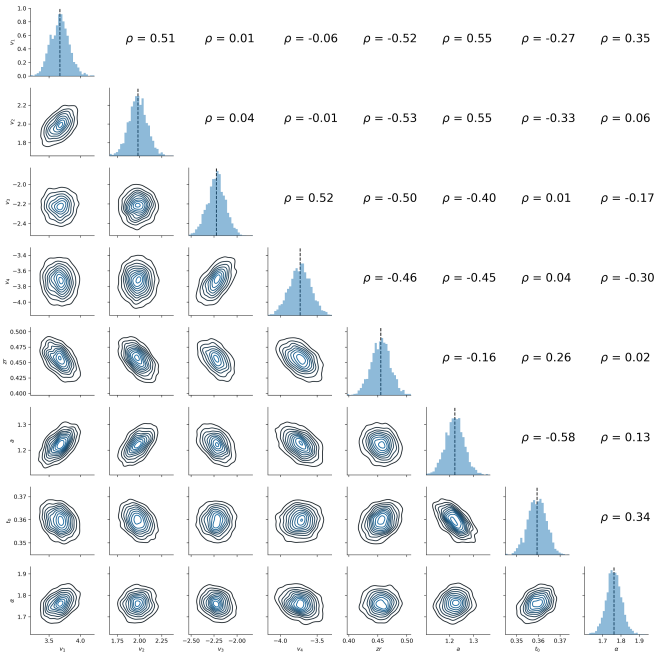


Figure 3: Example full posteriors and bivariate posterior correlations obtained from data of one participant in the long LDT via amortized Bayesian inference. Dashed lines on the main diagonal indicate posterior means.

We visualize the results by plotting the average R^2 metric obtained from fitting the LFM model to 300 simulated participants at different N between 50 and 1000 (see Figure 2a). Notably, recovery of the ground-truth parameters via posterior means is nearly perfect at higher trial numbers.

As a validation tool for visually detecting systematic biases in the approximate posteriors, we can also cheaply apply simulation-based calibration (SBC) and inspect the rank statistic of the posterior samples for uniformity (Talts, Betancourt, Simpson, Vehtari, & Gelman, 2018). Results from applying SBC to 5000 simulated participants at $N = 800$ are depicted in Figure 2b. Indeed, we confirm that no pronounced issues across all marginal posteriors are present.

Real Data Example

We can also apply the same networks from the previous simulation example for fully Bayesian inference on real data. Here, we fit the LFM model to previously unpublished data from eleven participants performing a long ($N = 800$ per condition) lexical decision task (LDT). Since the task had a 2×2 design, with a factor for *difficulty* (hard vs. easy), and a factor for *stimulus type* (word vs. non-word), we can assume a different drift rate for each design cell.

Applying the pre-trained networks, we immediately obtain samples from a full posterior over model parameters for each participant. Using the estimated posteriors, we can then test hypotheses about particular parameter values, compute individual differences, or compare means between conditions in a Bayesian way. Furthermore, we can analyze posterior correlations at an individual level and investigate task-dependent relationships between the α parameter and other parameters (see Figure 3 for results obtained from a single participant).

Across participants, α displays only small posterior correlations with drift rates as well as small posterior correlations with threshold and non-decision time parameters (mean $r < 0.5$ across all parameters of the standard diffusion model). These results provide first evidence that the α parameter can indeed be decoupled from other model parameters and possibly indicates a separate decision process.

Since the goal of this application was solely to illustrate a typical use case for amortized Bayesian inference, future research should focus on extensive external validation of the LFM model as well as proposing a neurocognitively plausible interpretation for the α parameter.

Outlook

The purpose of this work was to introduce the main ideas behind amortized Bayesian inference methods for simulation-based parameter estimation and model comparison. Although these methods come with promising theoretical guarantees and clear practical advantages, their utility for cognitive modeling is just beginning to be explored. Moreover, there are still many open questions and avenues for future research.

First, a systematic investigation of a potential *amortization gap* in certain practical application seems warranted. An amortization gap refers to a drop in estimation accuracy due

to the fact that we are relying on a single set of neural network parameters for solving an inference problem globally, instead of performing per-dataset optimization. Even though we have not observed such a scenario in our applications and simulations, this behavior might occur when the neural network estimators are not expressive enough to represent complex posterior distributions.

Second, there are still little systematic guidelines on how to best design and tune the neural network architectures so as to perform optimally across a variety of parameter estimation and model comparison tasks. Even though neural density estimation methods outperform standard ABC methods on multiple metrics and in various contexts, there is certainly room for improvement. Black-box optimization methods for hyperparameter tuning, such as Bayesian optimization or active inference (Snoek, Larochelle, & Adams, 2012), might facilitate additional performance gains and reduce potentially suboptimal architectural choices.

Finally, user-friendly software for applying Bayesian amortization methods *out-of-the-box* is still largely in its infancy. Developing and maintaining such software is a crucial future goal for increasing the applicability and usability of novel simulation-based methods.

Conclusion

We hope that the inference architectures discussed in this work will spur the interest of cognitive modelers from various domains. We believe that such architectures can greatly enhance model-based analysis in cognitive science and psychology. By leaving subsidiary tractability considerations to powerful end-to-end algorithms, researchers can focus more on the task of model development and evaluation to further improve our understanding of cognitive processes.

Acknowledgments

This research was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation; grant number GRK 2277 "Statistical Modeling in Psychology"). We thank the Technology Industries of Finland Centennial Foundation (grant 70007503; Artificial Intelligence for Research and Development) for partial support of this work.

References

Cranmer, K., Brehmer, J., & Louppe, G. (2019). The frontier of simulation-based inference. *arXiv preprint arXiv:1911.01429*.

Fontanesi, L., Gluth, S., Spektor, M. S., & Rieskamp, J. (2019). A reinforcement learning diffusion decision model for value-based decisions. *Psychonomic bulletin & review*, *26*(4), 1099–1121.

Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, *102*(477), 359–378.

Greenberg, D. S., Nonnenmacher, M., & Macke, J. H. (2019). Automatic posterior transformation for likelihood-free inference. *arXiv preprint arXiv:1905.07488*.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge university press.

Le, T. A., Baydin, A. G., & Wood, F. (2016). Inference compilation and universal probabilistic programming. *arXiv preprint arXiv:1610.09900*.

Lynch, S. M., & Western, B. (2004). Bayesian posterior predictive checks for complex models. *Sociological methods & research*, *32*(3), 301–335.

Marin, J.-M., Pudlo, P., Estoup, A., & Robert, C. (2018). *Likelihood-free model choice*. Chapman and Hall/CRC Press Boca Raton, FL.

Mestdagh, M., Verdonck, S., Meers, K., Loossens, T., & Tuerlinckx, F. (2019). Prepaid parameter estimation without likelihoods. *PLoS computational biology*, *15*(9), e1007181.

Myung, J. I., Montenegro, M., & Pitt, M. A. (2007). Analytic expressions for the bcdmem model of recognition memory. *Journal of Mathematical Psychology*, *51*(3), 198–204.

Radev, S. T., D'Alessandro, M., Bürkner, P.-C., Mertens, U. K., Voss, A., & Köthe, U. (2020). Amortized bayesian model comparison with evidential deep learning. *arXiv preprint arXiv:2004.10629*.

Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., & Köthe, U. (2020). Bayesflow: Learning complex stochastic models with invertible neural networks. *arXiv preprint arXiv:2003.06281*.

Ratcliff, R., Thapar, A., Gomez, P., & McKoon, G. (2004). A diffusion model analysis of the effects of aging in the lexical-decision task. *Psychology and aging*, *19*(2), 278.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959).

Stout, J. C., Busemeyer, J. R., Lin, A., Grant, S. J., & Bonson, K. R. (2004). Cognitive modeling analysis of decision-making processes in cocaine abusers. *Psychonomic bulletin & review*, *11*(4), 742–747.

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., & Gelman, A. (2018). Validating bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*.

Turner, B., Sederberg, P., & McClelland, J. (2016). Bayesian analysis of simulation-based models. *Journal of Mathematical Psychology*, *72*, 191–199.

Turner, B. M., Forstmann, B. U., Steyvers, M., et al. (2019). *Joint models of neural and behavioral data*. Springer.

Voss, A., Lerche, V., Mertens, U., & Voss, J. (2019). Sequential sampling models with variable boundaries and non-normal noise: A comparison of six models. *Psychonomic bulletin & review*, *26*(3), 813–832.

Wieschen, E. M., Voss, A., & Radev, S. (2020). Jumping to conclusion? a lévy flight model of decision making. *TQMP*, *16*(2), 120–132.

The Power of Nonmonotonic Logics to Predict the Individual Reasoner

Marco Ragni (ragni@cs.uni-freiburg.de)
Cognitive Computation Lab, University of Freiburg, Germany

Francine Wagner (wagner.francine94@gmail.com)
Cognitive Computation Lab, University of Freiburg, Germany

Sara Todorovikj (sara.todorovik@gmail.com)
Cognitive Computation Lab, University of Freiburg, Germany

Abstract

Human reasoning systematically deviates from conclusions predicted by classical logic. It is *nonmonotonic* and *defeasible*, i.e., new information can lead to the retraction of previous inferences. While these results hold for the analysis of population data, it is an open question, if nonmonotonic logics can capture individual human reasoning. In this article, we take three prominent nonmonontontic approaches, the Weak Completion Semantics, Reiters Default Logic, and OCF, a ranking on possible worlds, implement variants of them and evaluate them within the CCOBRA-framework for their predictive capability in the Suppression Task. We demonstrate that nonmonotonic approaches are able to predict individual reasoner on 82% (median). Furthermore, we can demonstrate that OCF and an improved version of Reiter make identical predictions and that abduction is relevant on the level of an individual reasoner. We discuss implications of logical systems for human reasoning.

Keywords: Human Reasoning; Nonmonotonic Logic; Evaluation; Individual Reasoner

Introduction

Humans draw different conclusions when they are presented with additional information. Consider the following information:

> If Lisa has an essay to write then she will study late in the library.
> She has an essay to write.

In a study about 96% of participants concluded, that *she will study late in the library* (Byrne, 1989). However, only 38% of the participants receiving in the same study the additional conditional

> If the library stays open, she will study late in the library.

endorsed the conclusion that *She will study late in the library*. In contrast to about 90% of the reasoners endorsing the respective conclusion when receiving the alternative conditional

> If she has a textbook to read, she will study late in the library.

While this change in the inference behavior between the first case to the second case might be intuitively clear for the reader, because the second conditional provides a possible constraint, the inference *she will study late in the library* even with such an additional constraint is classically logically valid. Hence, this and other findings (e.g., in the Wason Selection Task, see Ragni, Kola, & Johnson-Laird, 2018) demonstrate that the normative framework of classical logic is not a good descriptive framework for about *how* humans reason. Despite that logical inference problems require that humans derive the logical conclusion. In recent years this has lead to a number of different modeling approaches, e.g., probabilistic models (Oaksford & Chater, 2013), heuristic models (Evans & Over, 2004), and recently to the discovery that nonmonotonic logics might be appropriate (Stenning & Lambalgen, 2008). A reasoning theory is called nonmonotonic if new information can lead to the retraction of previous inferences (Antoniou, 1997). Hence, the reasoning process allows for defeasible conclusions or reasoning under exceptions. In fact, many cognitive theories are nonmonotonic, e.g., probabilistic (Oaksford & Chater, 2013), mental models (Johnson-Laird, Girotto, & Legrenzi, 2004), and cognitive logics (Stenning & Lambalgen, 2008). In this article, we investigate three noteworthy logical theories are to predict an individual reasoner conclusion, before the reasoner generates it. The paper is structured as follows: First, we will introduce the necessary background for conditional reasoning and the Suppression Task. In the next section we introduce three prominent models of nonmonotonic reasoning. Then, a section with the evaluation framework CCOBRA and the description of a data-set follows. Finally, a section on the evaluation of the different nonmonotonic logics and their variants and a discussion about the implications concludes the paper.

Background & Related Work

Conditional Reasoning

Conditional reasoning (i.e., reasoning about "if") is diverse from reasoning about given facts. It can represent assumptions about states, e.g., about causal relations or in action planning, considering hypothetically potentially different past states (e.g., counterfactual reasoning), or hypothesizing theories (e.g., inductive reasoning). It is highly relevant for both automated and human reasoning. There is a long history in cognitive science about modeling conditional reasoning, i.e., a statement of the form, "if e then l", often written by $e \rightarrow l$ or $(l|e)$. For a given conditional four inference mechanisms are possible:

Modus ponens (MP) is a deductively valid argument in classical logic: from the premises $e \rightarrow l$ and e, the consequent l is inferred. Consider the premises:

> If she has an essay to write then she will study late in the library. She has an essay to write.

The valid conclusion is:

> She will study late in the library.

Modus tollens (MT) is a deductively valid argument in classical logic: from the premises $e \rightarrow l$ and $\neg l$ the negated antecedent $\neg e$ is inferred. An example:

> If she has an essay to write then she will study late in the library. She will not study late in the library.

The valid conclusion is:

> She does not have an essay to write.

Denial of the antecedent (DA) is a deductively invalid argument in classical logic: from the premises $e \rightarrow l$ and $\neg e$ the negated consequent $\neg l$ is inferred. An example:

> If she has an essay to write then she will study late in the library. She does not have an essay to write.

The erroneous conclusion is:

> She will not study late in the library.

Affirmation of the consequent (AC) is also a deductively invalid argument in classical logic: from the premises $e \rightarrow l$ and l, the antecedent e is inferred. An example:

> If she has an essay to write then she will study late in the library. She will study late in the library.

The erroneous conclusion is:

> She has an essay to write.

All inference mechanisms are logically valid in case of a biconditional interpretation, i.e., if and only if she has an essay to write she will study late in the library. It has been claimed that the nonmonotonic System P satisfies minimal rationality criteria. And, that it is close to human reasoning (Pfeifer & Kleiter, 2005). A different study could not support the relevance of the nonmonotonic System P as a good descriptive theory to explains psychological findings (Kuhnmünch & Ragni, 2014). So we exclude this theory.

Suppression Task

Together with the aforementioned Wason Selection Task, the Suppression Task (the difference in resoning behavior with or without the additional conditional) is one core problem for reasoning theories (Byrne, 1989; Neth & Beller, 1999; Chan & Chua, 1994; Politzer, 2005). Accordingly, this task and human nonmonotonic reasoning has been modeled by many researchers (Dietz, Hölldobler, & Ragni, 2012a; Stenning & Lambalgen, 2008). Recent research (Ragni, Eichhorn, & Kern-Isberner, 2016) analyzed if nonmonotonic systems have the competence to grasp the specific nonmonotonicty of the Suppression Task without additional background information. It demonstrated that the inference mechanisms of all nonmonotonic logics despite the weak completion semantics required additional knowledge. This was, however, an evaluation on the aggregated level. So theories and models were competent to solve the problems with some requiring additional background knowledge. So far no analysis on predictive performance of the cognitive models or logics for the individual human reasoner on conditional reasoning in the Suppression Task. Before we can do so we present three main theories for nonmonotonic reasoning.

Models of Nonmonotonic Reasoning

The Weak Completion Semantics

One main criticism against classical two valued approaches is that in everyday life we typically have degrees of (un-) certainty. The traditional two symbols $\top$, $\bot$, are extended with U denoting *true*, *false*, and *unknown*, respectively. Stenning and Lambalgen (2008) have claimed that conditionals should be encoded by "licenses for inferences". For example, the conditional *if she has an essay to finish, she will study late in the library* or short ($l \leftarrow e$) should be encoded by the clause $l \quad e \wedge \overline{ab}_1$, where ab_1 is an *abnormality* predicate which expresses that l holds if e holds and nothing abnormal is known. The programs obtained for the two main examples of the Suppression Task are depicted in the first two columns of Table 1.

The abnormality predicates (e.g., ab_1) represent abnormal cases: For instance, ab_1 is true when *the library does not stay open* and ab_3 is true when *she does not have an essay to finish*. Weak completion is the process of substituting the conditional with a biconditional.

In the case of AC where the conclusion holds the propositional variable e is mapped to unknown. Hence, if we observe that 'she will study late in the library', then we cannot explain by the model that 'she has an essay to write' without abduction (Saldanha, Hölldobler, & Rocha, 2017). Abduction searches for the minimal explanation. Since e is the only undefined propositional letter in this context, the set of abducibles is $e \leftarrow \top$, $e \leftarrow \bot$. The above observation can be explained by selecting $e \leftarrow \top$ from the set of abducibles, weakly completing it to obtain $e \leftarrow \top$ and adding this equivalence to the logic program.

Reiter's Model for Default Reasoning

Reiter (1980) proposed a system for default reasoning. According to Reiter, conditionals of the form $e \rightarrow l$ are interpreted as default rules, i.e., they are true as long as no exception is known. This idea was inspired that the conditional "if an animal is a bird, then it can fly" is true as long as we know that this animal is not a penguin (or any other exception such as a dodo etc.). For reasons of simplicity we do not introduce the formalizations of a background theory. A default rule constructed from a conditional has:

Table 1: The WCS approach to the suppression task. ELT = the statements 'essay', 'late', and 'textbook to read'; ELO = the statements 'essay', 'late', 'open hold'; *ab* are abnormality predicates; wc is the weak completion; and lm_L are the least model; Table adapted from Dietz et al. (2012b).

Premise	$\mathcal{P}_{ELT}$	$\mathcal{P}_{ELO}$
Clauses	$l \leftarrow e \wedge \overline{ab}_1$ $l \leftarrow t \wedge \overline{ab}_2$ $ab_1 \leftarrow \bot$ $ab_2 \leftarrow \bot$ $e \leftarrow \top$	$l \leftarrow e \wedge \overline{ab}_1$ $l \leftarrow o \wedge \overline{ab}_3$ $ab_1 \leftarrow \overline{o}$ $ab_3 \leftarrow \overline{e}$ $e \leftarrow \top$
$\mathsf{wc}\,\mathcal{P}$	$l \leftrightarrow (e \wedge \overline{ab}_1)$ $\vee (t \wedge \overline{ab}_2)$ $ab_1 \leftrightarrow \bot$ $ab_2 \leftrightarrow \bot$ $e \leftrightarrow \top$	$l \leftrightarrow (e \wedge \overline{ab}_1)$ $\vee (o \wedge \overline{ab}_3)$ $ab_1 \leftrightarrow \overline{o}$ $ab_3 \leftrightarrow \overline{e}$ $e \leftrightarrow \top$
$\mathsf{lm}_L\mathsf{wc}\,\mathcal{P}$	$\langle\{e,l\},\{ab_1,ab_2\}\rangle$	$\langle\{e\},\{ab_3\}\rangle$
$\mathsf{lm}_L\mathsf{wc}\,\mathcal{P}$	Infers l	Not infers l
Empirical Results Byrne (1989)	96 % infer l	38 % infer l

$$\frac{e : \textit{justifications}}{l}$$

- the precondition: e (an *e*ssay to write)
- the justifications: depends on the scenario, for the library scenario a justification is that the library is open.
- the consequence: l (study *l*ate in the library)

To model the Suppression Task we construct the default rules from the conditionals, i.e., for the conditional "if Lisa has an essay to write then she will study late in the library." we assume that no exception is known and so the exception above is empty. This changes when we learn about the exception that the library is not open, then the justification is that the library is open. Facts can be formulated as a conditional with a true antecedent."Lisa has an essay to finish"

$$\frac{:}{e}$$

This means that without any precondition and without any justification it can be inferred that Lisa has an essay to finish. Conditionals with tautology as antecedent are plausible statements or facts about the world (Beierle & Kutsch, 2019). Statements such as "Mostly, Lisa has an essay to finish" can be translated to:

$$\frac{: e}{e}$$

Table 2: The successive generation of ranks (ranks are represented by κ) of possible worlds for the conditionals 'if e then l' and 'if t then l'.

	e	l	t	$\kappa(\omega)$	$\omega_i \models (l\|e)$	$\kappa(\omega)$	$\omega_i \models (l\|t)$	$\kappa(\omega)$
ω_1	0	0	1	0	$\overline{e}$	1	$t\overline{l}$	3
ω_2	0	1	1	0	$\overline{e}$	1	tl	1
ω_3	1	0	1	0	$e\overline{l}$	2	$t\overline{l}$	4
ω_4	1	1	1	0	el	0	tl	0

the justification ensures that the cases where she does not have an essay to finish, are handled as an exception. Our implementation works as follows: After translating the fact and conditionals, the defaults are executed, starting with the fact and keeping the order of the conditionals from the original task. Since the defaults, constructed from the fact, do not have any precondition and our knowledge about the world (which is written as W in Reiter's terminology) is empty, they are always added to the world knowledge W.

Ordinal conditional function

A different approach is inspired by the relevance of worlds and so some worlds are more relevant than others. This inspired the idea that the relevance of the worlds impose a rank on the worlds. There are three kinds of worlds for a conditional $(l|e)$:

- worlds satisfying e and l, $\omega \models el$
- worlds falsifying the conditionals, assuming e is true but the consequence l to be false $\omega \models e\overline{l}$
- worlds assuming e to be false, $\overline{e}$, called inapplicable worlds.

An inapplicable conditional means that there can't be made any statement about l, as e is already assumed to be false. The different sets can be represented by an indicator function (Calabrese, 1991):

$$(l|e)(\omega) = \begin{cases} 1 & \textit{If } \omega \models el \\ 0 & \textit{If } \omega \models e\overline{l} \\ u & \textit{If } \omega \models \overline{e} \end{cases}$$

where u means undefined, i.e., a case where the precondition is false. This represents that when the precondition does not hold, a conclusion about the truth value of the consequence relation l cannot be made. The conditional $(l|e)$ is evaluated as true (has the value 1), if the possible world ω has e and l as true. In this case we write for this world el.

Instead of assigning probabilities to a world, we can use ordinal conditional functions (OCFs)

$$\kappa : \Omega \to \mathbb{N} \cup \{\infty\} \textit{ with } \kappa^{-1}(0) = \emptyset$$

which maps possible worlds to an integer value. They map any possible world $\omega \in \Omega$ to natural number, which represents the degree of disbelief (Spohn, 1988). They express degrees of plausibility of propositional formulas ϕ by specifying degrees of disbelief of their negation $\bar{\phi}$ (Beierle & Kutsch, 2019). The more plausible a world ω is, the smaller its rank $\kappa(\omega)$. Consequently $\kappa(\omega_1) = 0$ describes the world ω_1, which is the most plausible. Hence $\kappa(\omega_2) = 1$ is a world a little bit less plausible, than ω_2, whereas $\kappa(\omega) = 10$ or higher is a world very unplausible. At least one world needs to have the ranking of 0. $\kappa \models (l|e)$, *iff* $\kappa(el) < \kappa(e\bar{l})$ means the agent would conclude that the verification of the conditional is more plausible or less surprising than the falsification of the consequence (Beierle & Kutsch, 2019). The main idea of our model is that all possible worlds or assignments, are ranked by their plausibility. The most promising world, that matches our choices will be returned accordingly.

An example for computing ranks of worlds For every task, we assume that, in the beginning, all worlds are equally possible and have the rank 0 (cp. Table 2), since we do not know anything about our environment. The rank will be updated with multiple conditionals. Secondly all κ values for these sets are determined: $\kappa(el)$, $\kappa(e\bar{l})$, $\kappa(\bar{l})$. The κ values for the sets will have the same rank as the world with the smallest rank from each set accordingly. In Table 1 we have an example of possible worlds, which are revised with two conditionals. For the input:

> If Lisa has an essay to finish, she will study late in the library. She will study late in the library.

The possible worlds can describe an environment with 2 literals l and e. We write $\top$ for a tautology, i.e., the truth value *true*. The first sentence of the task is encoded to the conditional $(l|e)$ and the fact is encoded to a conditional without a precondition $(l|\top)$.

At the beginning all worlds are equally plausible, therefore $\kappa(\omega) = 0$. We use the first conditional $(l|e)$ to revise the belief and to update the κ values for each world. Therefore the worlds are split into three sets: verifying, falsifying and inapplicable worlds. Then, the variables $\kappa(el)$, $\kappa(e\bar{l})$, $\kappa(\bar{e})$ can be determined, all having a starting value 0. Finally, all worlds are updated accordingly and a new conditional can revise our belief in the same way the first one did. The second information is a fact and is transformed into a conditional with a precondition which is always true: $(l|\top)$. The values for $\kappa(el)$, $\kappa(e\bar{l})$, $\kappa(\bar{e})$ are computed. The most plausible world is the last one. When having the choice between:

> Lisa has an essay to finish
> Lisa has not an essay to finish

we will chose the first one, because it is consistent with the most plausible world: ω_4.

Individual Predictions

CCOBRA

The Cognitive Computation for Behavioral Reasoning Analysis (CCOBRA) framework is a benchmarking tool implemented in Python that actively integrates the individual human into the prediction loop. There is a close connection to psychological experiments. Implemented models are supposed to simulate the experimental procedure for individual participants. By providing responses to individual tasks, models are evaluated based on their predictive accuracies[1]. The CCOBRA framework offers multiple possibilities, e.g., a pretrain, adapt and predict methods that we used for our model evaluation.

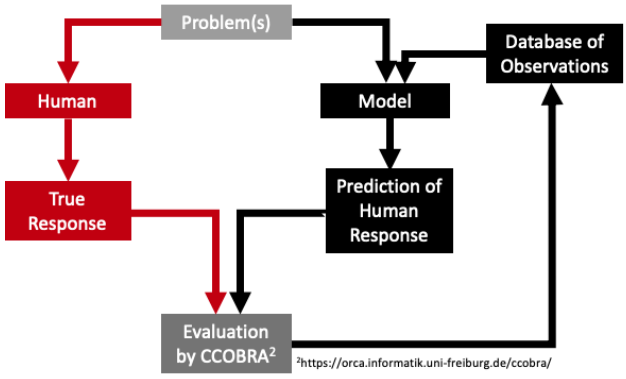


Figure 1: The CCOBRA-framework to evaluate the predictive power of cognitive theories.

Data-Set

The data can be found online[2]. It consists of 96 participants with no background in logic. Participants were recruited for a laboratory experiment at the University Freiburg. In 12 problems participants were requested to answer if specific conclusions follow from given information. Participants were divided into four groups. Group A received tasks with simple conditional arguments (non-suppression group). Group B also received tasks with simple conditional arguments but with a linguistic modification in premise one by adding the keyword "mostly" (they received the problem description in German).

> If Lisa has an essay to finish then she will mostly study late in the library. Lisa will study late in the library.
> Does Lisa have an essay to finish?

Group C received the modification in premise two by adding the keyword "mostly":

> If Lisa has an essay to finish then she will study late in the library. Mostly, Lisa will study late in the library.
> Does Lisa have an essay to finish?

[1] http://orca.informatik.uni-freiburg.de/ccobra

[2] https://github.com/CognitiveComputationLab/cogmods/tree/master/suppression_task

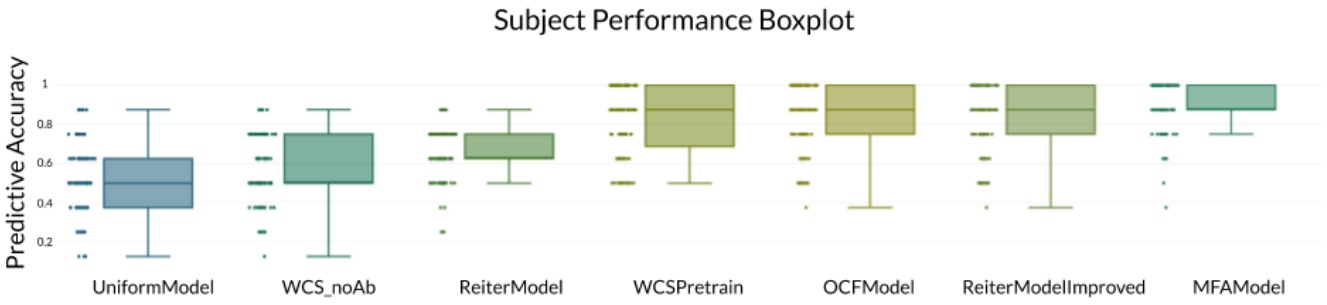


Figure 2: Boxplots for the models indicating individual subject performance. The data used for the plot are accuracies for each individual human reasoners performance (cp. Fig. 1.)

Group D received items with simple conditional arguments and two additional arguments:

> If Lisa has an essay to finish then she will study late in the library. If Lisa has some textbooks to read then she will study late in the library. If the library stays open then she will study late in the library. Lisa will study late in the library. Does Lisa have an essay to finish?

All four groups received three different scenarios (an abstract version, a story on an exoplanet with aliens having specific properties, and the library example above) and these in turn with the four inferential figures modus ponens (MP), modus tollens (MT), denial of the antecedent (DA) and affirmation of the consequent (AC). The arguments were presented sequentially and the concluding question had to be answered with "yes" or "no".

As our goal is to model the individual reasoner, we only report some aggregate statistic: Group A indicated the highest likeliness (88%) that the made conclusion holds true followed by Group D (80%), Group C (70%) and Group B (67%). For the logical correctness Group D shows the highest correctness rate (61.0%) followed by Group C (55.8%), Group A (54.9%) and then Group B (52.0%).

Predictive Performance of the Models

We compare now the predictive performance of the models with each other and baselines (cp. Fig. 2):

Baseline 1: Most Frequent Answer. To compare the cognitive models the most frequent answer is a good empirical measure. It represents the answers of all participants given for the same problem and returns the response which was answered the most. Taking the majority vote into account, it achieves a precision of 89.2%. This means that participants are quite homogeneous, i.e., they do not differ much in the responses they give.

Baseline 2: The Uniform Model. The *uniform random model* provides another base line, namely for participants that select randomly an answer. In the presented experiments, the participants had two choices for each task. So the uniform random model selects randomly one of the two choices and returns it as the predicted answer. It achieved, as expected, a 50% prediction performance.

The Weak Completion Semantics. This model which is based on a ternary logic and logic programs with allowing for abductive reasoning has a predictive performance of 82% based with an upper bound of 100% and a lower bound of 50%. If abductive reasoning is not allowed the performance drops to 56.1%, with a lower 12.5% and an upper bound of 87.5%.

The OCF-Model. The OCF model which is based on computed ranks of models reaches a high level of predictivity of about 82.2%. This model achieved an accuracy of 82.2%, with an upper bound of 100% and a lower bound of one single person with 37.5%.

The Classical Reiter Model. This is Reiter's original model Reiter (1980). It achieves a predictive accuracy of 65.5%, with predicting some persons as high as 87.5% and others as low as 25%.

Reiter Model Improved. The basic Reiter Model can be extended by adding default rules in order to model the phenomen that subject tend to use the modus tollens or affirmation of the consequent inference rule. By adding these two rules we reach the identical predictivity of the OCF-model with 82.2%, i.e., it predicts the exact same answer for every single subjects. This demonstrates a functional equivalence of the Reiter Model that is augmented with two additional rules with the semantic based approach by the OCF.

Discussion

Human reasoning has often been disqualified as "unlogical". While many psychological findings demonstrate that humans do deviate from valid inferences predicted by classical logic,

this paper demonstrates that nonmonotonic logics are competitive. They are even able to predict a median of 82% of the inferences drawn by every individual human reasoner in the Suppression Task. While the classical approach by Reiter did not match the high performance of the OCF model, we extended Reiter's model with two rules and demonstrated a functionally equivalent model to the OCF. The performance of the pre-trained version of the WCS model is only slightly lower than Reiter Model Improved and OCF Model. WCS deviated on some problems in the MP-case from the participants responses, where due to introduced abnormalities, it did not predict the classical MP conclusion. On the other hand, the WCS model successfully managed to model Denial of Antecedence problems by abductive reasoning in the cases of induced non-monotonicity, which the Reiter Model Improved and OCF did not succeed in. This analysis gives further support for the importance of abductive reasoning that has been reported relevant in the Weak Completion Semantics (Breu, Ind, Mertesdorf, & Ragni, 2019). Focusing on the individual predictivity of each system in a training set and a test set of participants in the CCOBRA-framework allows to estimate the true power of logics and cognitive models and makes even more progress possible, because it allows to identify individuals that are perfectly predicted and individuals in turn that are not captured. Future research needs to cover more experimental data, more cognitive theories, and aim to identify successful mechanisms of highly-predictive theories.

Acknowledgements

This paper was supported by a DFG-Heisenbergfellowship and DFG grants RA 1934/9-1, RA 1934/4-1, and RA 1934/3-1 to MR.

References

Antoniou, G. (1997). *Nonmonotonic reasoning*. Cambridge, MA, USA: MIT Press.

Beierle, C., & Kutsch, S. (2019). Systematic generation of conditional knowledge bases up to renaming and equivalence. In *European conference on logics in artificial intelligence* (pp. 279–286).

Breu, C., Ind, A., Mertesdorf, J., & Ragni, M. (2019). The weak completion semantics can model inferences of individual human reasoners. In *Logics in artificial intelligence - 16th european conference, JELIA 2019, rende, italy, may 7-11, 2019, proceedings* (pp. 498–508).

Byrne, R. (1989). Suppressing valid inferences with conditionals. *Cognition*, *31*, 61–83.

Calabrese, P. (1991). *Deduction and inference using conditional logic and probability* (Tech. Rep.). Naval Ocean Systems Center San Diego CA.

Chan, D., & Chua, F. (1994). Suppression of valid inferences: syntactic views, mental models, and relative salience. *Cognition*, *53*(3), 217 - 238.

Dietz, E.-A., Hölldobler, S., & Ragni, M. (2012a). A Computational Approach to the Suppression Task. In N. Miyake, D. Peebles, & R. Cooper (Eds.), *Proceedings of the 34th Annual Conference of the Cognitive Science Society* (pp. 1500–1505). Austin, TX: Cognitive Science Society.

Dietz, E.-A., Hölldobler, S., & Ragni, M. (2012b). A Simple Model for the Wason Selection Task. In T. Barkowsky, M. Ragni, & F. Stolzenburg (Eds.), *Report Series of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition.*

Evans, J. S. B. T., & Over, D. E. (2004). *If*. Oxford: Oxford University Press.

Johnson-Laird, P. N., Girotto, V., & Legrenzi, P. (2004). Reasoning from inconsistency to consistency. *Psychological Review*, *111*(3), 640.

Kuhnmünch, G., & Ragni, M. (2014). Can Formal Nonmonotonic Systems Properly Describe Human Reasoning? In P. Bello, M. Guarini, M. McShane, & B. Scassellati (Eds.), *Proceedings of the 36th annual conference of the cognitive science society* (p. 1222-1228). Austin, TX: Cognitive Science Society.

Neth, H., & Beller, S. (1999). How knowledge interferes with reasoning-suppression effects by content and context. In *Proceedings of the twenty first annual conference of the cognitive science society* (pp. 468–473).

Oaksford, M., & Chater, N. (2013). Dynamic inference and everyday conditional reasoning in the new paradigm. *Thinking & Reasoning*, *19*(3-4), 346-379.

Pfeifer, N., & Kleiter, G. D. (2005). Coherence and nonmonotonicity in human reasoning. *Synthese*, *146*(1-2), 93–109.

Politzer, G. (2005). Uncertainty and the suppression of inferences. *Thinking & Reasoning*, *11*(1), 5-33.

Ragni, M., Eichhorn, C., & Kern-Isberner, G. (2016). Simulating Homan Inferences in the Light of New Information: A Formal Analysis. In *Proceedings of the 25th international joint conference on artificial intelligence (ijcai'16).*

Ragni, M., Kola, I., & Johnson-Laird, P. N. (2018). On selecting evidence to test hypotheses: a theory of selection tasks. *Psychological Bulletin*, *144*(8), 779-796.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, *13*(1-2), 81–132.

Saldanha, E.-A. D., Hölldobler, S., & Rocha, I. L. (2017). The weak completion semantics. In *Bridging@ cogsci* (pp. 18–30).

Spohn, W. (1988). Ordinal conditional functions: A dynamic theory of epistemic states. In *Causation in decision, belief change, and statistics* (pp. 105–134). Springer.

Stenning, K., & Lambalgen, M. (2008). *Human reasoning and cognitive science*. MIT Press.

Using cross-validation to determine dimensionality in multidimensional scaling

Russell Richie (drrichie@sas.upenn.edu)
Department of Psychology, Philadelphia, PA, USA

Steven Verheyen (verheyen@essb.eur.nl)
Department of Psychology, Rotterdam, Belgium

Keywords: similarity; multidimensional scaling; cross-validation; model selection; dimensionality

Introduction

Multidimensional scaling (MDS) is a popular technique for embedding items in a low-dimensional spatial representation from a matrix of the dissimilarities among items (Shepard, 1962). MDS has been used simply as a visualization aid or dimensionality reduction technique in statistics and machine learning applications, but in cognitive science, MDS has also been interpreted as a cognitive model of similarity perception or similarity judgment, and is often part of a larger framework for modeling complex behaviors like categorization (Nosofsky, 1992) or generalization (Shepard, 2004). However, a persistent challenge in application of MDS is selecting the latent dimensionality of the inferred spatial representation; the dimensionality is a hyperparameter that the modeler must specify when fitting MDS.

Perhaps the most well-known procedure for selecting dimensionality is constructing a scree plot of residual stress (the difference between empirical dissimilarities and dissimilarities implied by the model) as a function of dimensionality, and then looking for an elbow: the dimensionality where stress has decreased dramatically but then plateaus. This elbow is taken to indicate that extending the space with additional dimensions does not substantially improve the fit of the model to the input similarities. Unfortunately, this procedure is highly subjective. Often such elbows do not exist, and instead the scree plots show a smooth decrease in stress as MDS increasingly overfits to noise at higher dimensionalities. In response, various more principled statistical techniques for model selection have been proposed that account for the trade-off between model complexity (dimensionality) and model fit (stress), including likelihood ratio tests (Ramsay, 1977), BIC (Lee, 2001), and Bayes factors (Gronau and Lee, in press). While such techniques are valuable, they can be prohibitively computationally complex for novice MDS users, and rely on a number of assumptions that are not necessarily met (e.g., Storms, 1995).

An alternative technique that may avoid such problems is cross-validation. Under this approach, MDS of a given dimensionality would be fit to some subset of available dissimilarity data, the model's predicted distances for held-out dissimilarity data would be evaluated, and the dimensionality which maximizes performance on the held-out data would be selected. Despite the simplicity and generality of cross-validation as a model selection procedure, cross-validation has seen relatively little application to MDS or related methods (Steyvers, 2006; Roads & Mozer, 2019; Gronau & Lee, in press), with no systematic exploration of its capabilities, as there has been for likelihood ratio tests, BIC, and Bayes factors (Ramsay, 1977; Lee, 2001; Gronau & Lee, in press). In the present work, we therefore examine the usefulness of cross-validation over cells of a dissimilarity matrix in simulations and applications to empirical data.

Simulations

We conducted a standard model recovery exercise, whereby we simulated spaces of known dimensionality, from which we collected and aggregated noisy dissimilarity data, and applied cross-validation to attempt to recover the true dimensionality. Our simulations were conducted as follows:

1. Sample *n* items uniformly from the unit hypercube of dimensionality between 1 and 7
2. For each simulated subject, add noise ~ *N(0, sd)* to the item coordinates, and compute the inter-item Euclidean distances
3. Average over subject distance (dissimilarity) matrices
4. Derive a weight for each cell of the average distance matrix equal to the inter-subject precision of that cell
5. Generate 10 random 80-20 train-test splits of the averaged matrix such that each row of each training matrix is missing no more than 75% of its cells
6. For each train-test split:
    1. For each dimensionality from 1 to 7:
        1. Fit ratio MDS to the training dissimilarities and the cell weights given by (4), using the *smacof* library in R (de Leeuw & Mair, 2017)
        2. Use the fitted MDS to obtain distances for the cells of the test split
        3. Compute Pearson correlation between the MDS distances and held-out dissimilarities
7. Select the dimensionality with the highest median correlation across all train-test splits

Figure 1 shows distributions of best-fitting dimensionalities (y-axis) over 50 simulations of a particular true dimensionality (x-axis), number of subjects (hue), noise level (columns), and number of items (rows). Figure 1 shows the true dimensionality is recoverable across a range of conditions.

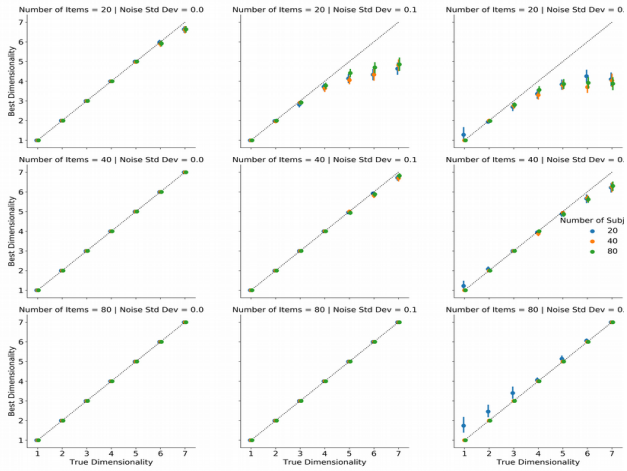


Figure 1. Error bars are 95% confidence intervals.

Empirical application

We applied steps 3-7 above to an empirical dataset of similarity judgments from Hout, Goldinger, and Ferguson (2013), who had 92 subjects use the Spatial Arrangement Method to judge similarity among a set of 27 artificial 'bugs' which varied on 3 dimensions (darkness, pincer shape, number of legs). Figure 2 shows distributions of Pearson correlations between MDS distances and held-out dissimilarities under 100 train-test splits for each fitted dimensionality from 1 to 7. Cross-validation correctly selects a 3-dimensional spatial representation for these data.

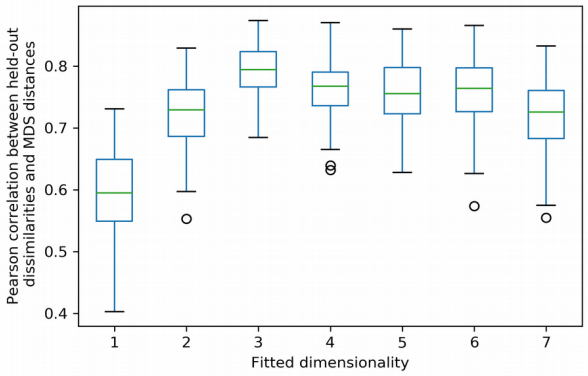


Figure 2

Discussion

We have demonstrated the utility of cross-validation for determining the dimensionality of multidimensional scaling models, given subject-averaged similarity data and assumptions (or knowledge) that dissimilarity data are on a ratio scale and were generated from a Euclidean distance metric. We cross-validated across individual cells of a dissimilarity matrix, whereas previous applications of cross-validation to MDS cross-validated over subjects (Steyvers, 2006). We believe our approach has certain advantages, e.g., it can be applied to single subject data, and might eventually be applicable in individual differences scaling, a direction we are now pursuing. This latter extension may be especially important, because averaging dissimilarity matrices might not always be warranted (Ashby, Maddox, & Lee, 1994). We are also currently exploring simulations and empirical applications when certain current constraints are relaxed, e.g., when similarity data are on a likert scale.

Acknowledgments

This work was funded by National Science Foundation grant SES-1626825, and the Alfred P. Sloan Foundation.

References

Ashby, F. G., Maddox, W. T., & Lee, W. W. (1994). On the dangers of averaging across subjects when using multidimensional scaling or the similarity-choice model. *Psychological Science, 5*(3), 144-151.

Gronau, Q.F., & Lee, M.D. (in press). Bayesian inference for multidimensional scaling representations with psychologically-interpretable metrics. *Computational Brain & Behavior.*

Hout, M. C., Goldinger, S. D., & Ferguson, R. W. (2013). The versatility of SpAM: A fast, efficient, spatial method of data collection for multidimensional scaling. *Journal of Experimental Psychology: General, 142*(1), 256–281. https://doi.org/10.1037/a0028860

Lee, M. D. (2001). Determining the dimensionality of multidimensional scaling representations for cognitive modeling. *Journal of Mathematical Psychology, 45* (1), 149–166.

de Leeuw, J., & Mair, P. (2009). Multidimensional Scaling Using Majorization: SMACOF in R. *Journal of Statistical Software, 31*(3), 1-30.

Nosofsky (1992). Similarity scaling and cognitive process models. *Annual Review of Psychology, 43* , 22 –53.

Ramsay, J. O. (1977). Maximum likelihood estimation in multidimensional scaling. *Psychometrika, 42*(2), 241-266. https://doi.org/10.1007/bf02294052

Roads, B. D., & Mozer, M. C. (2019). Obtaining psychological embeddings through joint kernel and metric learning. *Behavior Research Methods*. doi: 10.3758/s13428-019-01285-3

Shepard, R. N. (1962). The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika, 27*, 125–140.

Shepard, R. N. (2004). How a cognitive psychologist came to seek universal laws. *Psychonomic Bulletin & Review, 11*, 1-23. https://doi.org/10.3758/bf03206455

Steyvers, M. (2002). Multidimensional Scaling. In: *Encyclopedia of Cognitive Science*. Nature Publishing Group, London, UK.

Storms, G. (1995). On the robustness of maximum-likelihood scaling for violations of the error model. *Psychometrika, 60* (2), 247-258. https://doi.org/10.1007/BF02301415

A Computational Cognitive Model of Reasoning in Tibetan Buddhist Monastic Debate

Stefan Riegl (stefan.riegl@student.ru.nl)
Radboud University, Comeniuslaan 4
6525 HP, Nijmegen, The Netherlands

Marieke van Vugt (m.k.van.vugt@rug.nl)
University of Groningen, Nijenborgh 9
9747 AG Groningen, The Netherlands

Abstract

In Tibetan monasteries, the education system relies heavily on a very specific style of debating that is at once exhilarating and intellectually rigorous. Relatively little research has been done on the psychological and neural mechanisms of this debate, which may be an interesting method for education around the world. Hence the formation of a theory of this practice is important. Here we present a computational theory of Tibetan monastic debate implemented in the ACT-R cognitive architecture. We complement the ACT-R model with graph theory to represent knowledge and show how we can capture the dynamic flow of a debate in our model. Future research should validate the model in its native population and enrich it with more detailed strategies. Nevertheless, we think it provides an interesting example of how the interactive process of debating can be modelled.

Keywords: Reasoning; monastic debate; ACT-R; reasoning agents; graph-theory; logic; knowledge representation.

Introduction

Different cultures have different education methods. While Western education has been well studied, other forms of teaching and learning exist. One intriguing method is Tibetan monastic debate, a part of analytical meditation practises; a method that is practised in a dyad and is said to date back to the 10th century (van Vugt et al., 2019). It bears similarities to the Socratic method.

This method of debating is characterised by vigorous exercise, excitement, as well as focus on critical thinking and examining many different perspectives. In other words, a dialectic method like the monastic debate would introduce an inherently social aspect to studying, that would complement more individual-centred study approaches. With its game-like structure and active methods, debate may activate students and help them to think more critically.

Repeated practise of debate is likely to have substantial impact on the cognitive development of the debater. Recent research (van Vugt et al., 2020; van Vugt et al., 2019) speculated that debate has a positive effect on executive functions, such as e.g. critical thinking, emotion regulation, or social cognition. However, up until today no evidence has been found that debate trains cognitive functions. Further research is necessary to identify the cognitive skills that are at play during monastic debate.

Modelling debate helps drawing up specific hypotheses what cognitive skills are required for debate. With a model predictions can be made that certain cognitive skills could be improved after intense practice (Taatgen, 2013) and what can happen to debate if a certain skill is missing or not available due to e.g. exhaustion or speed pressure.

Monastic debate

Monastic debate is part of all Tibetan monastic traditions, but with several hours a day over the course of 20 years (Dreyfus, 2003) it is most intensively practised in the Geluk school. In general, monastic education is centred around the study and memorisation of Buddhist scriptures. Debate is used to test and deepen the debater's knowledge, and to sharpen skills in critical thinking and logical reasoning. The general method of debate can be understood as a form of *reductio ad absurdum* common in logical argumentation. What makes this debate form unique is that it aims at uncovering shortcomings and inconsistencies in in the debater's knowledge and understanding. This sets Monastic debate apart from other common debating styles, where debaters attempt to defeat the other debater with stronger arguments. An example of monastic debate can be seen in figure 1.

Debate is a dialogue between a "challenger" and a "defender" (Perdue, 2014) and typically covers topics of recently discussed lessons in Buddhist philosophy. Generally the challenger proposes statements to which the defender responds. The defender has to choose between accepting and rejecting a statement, while ensuring that no statement is accepted that contradicts an earlier one. Statements are intended to probe the consistency of a particular philosophical position of the other debater. However, they are not required to be rational or correct, as they can be understood as a tool to explore the consequences of adopting a particular philosophical position. Monastic debate follows a formal schema that includes choreographic elements like shouting and clapping, but also statement-response patterns, as outlined in table 1.

The structure of debate just discussed may lend it well as a tool for scientists. According to monks from the Sera Jey monastery, debate supports the investigation of a topic from various perspectives by exploring the consequences of adopting a particular position. So on the one hand, investigating the research hypotheses by means of debate can help to reveal latent, inconsistent or overly restricting assumptions, which can then be resolved. On the other hand debate is an interaction form that can promote novel ways of thinking about a topic and as such can lead to fresh insights into e.g. scientific

results. Moreover, excellent ownership of the material is required to keep up with the speed of debate and maintain its formal structure, which pushes the boundaries of scientists in an engaging way.

Previous research

Scientific research on the psychological and neural mechanisms of debate is very limited. The monastic curriculum (Liberman, 2007; Dreyfus, 2003) and debate as a whole (Perdue, 2014) were discussed. It was suggested that debate might share commonalities with the Socratic method or might be a "mode of inquiry" (Dreyfus, 2003). However there seems to be a paucity of research on what effects debate has on the individual, especially from a cognitive science perspective.

The single, empirical study that has so far been done (van Vugt et al., 2020) showed that mid-frontal theta oscillations (neural correlate for concentration) increased during debate. It also showed that frontal alpha oscillations synchronised between debaters when they agreed compared to when they disagreed. The researchers theorised that successful debate requires a rich set of cognitive skills and that repeated exercise over years fosters those skills. A conceptual model was proposed (van Vugt et al., 2019) that suggests how debate requires skills in focused attention, working memory, and logical reasoning, but also emotion regulation and mental flexibility. Further research could provide more evidence what particular cognitive processes are relevant and change during training in this method.

C: Dhih! The subject, in just the way *[Manjushri debated]*. Is whatever is a colour necessarily red?
D: I accept *[that whatever is a colour is necessarily red]*.
C: It follows that whatever is a colour is necessarily red.
D: I accept it.
C: It *[absurdly]* follows that the subject, the colour of a white religious conch, is red.
D: Why *[is the colour a white religious conch red]*?
C: Because of being a colour. You asserted the pervasion *[that whatever is a colour is necessarily red]*.
D: The reason *[that the colour of a white religious conch is a colour]* is not established.
C: It follows that the subject, the colour of a white religious conch, is a colour because of being white.
D: The reason *[that the colour of a white religious conch is white]* is not established.
C: It follows that the subject, the colour of a white religious conch, is white because of being one with the colour of a white religious conch.
D: I accept that the colour of a white religious conch is white.

Figure 1: An example of Tibetan Buddhist monastic debate, adapted from (Perdue, 2014). *C* denotes the challenger and *D* the defender.

Statement type, example (challenger)	**Possible responses (defender)**
Two-part debate "Red is a visual form"	• "I accept." • "Why?" / "No."
Three-part debate "Red is a visual form, because it is a colour."	• "I accept." • "Reason not established." • "No pervasion."
Inquire about reason "Red is a visual form, because…"	• "…because red is a colour."
Request an example (context-dependent) "Posit it!"	[Context: Something that is a bird and that cannot fly.] • "A penguin."

Table 1: Typical statements types of the challenger and possible responses of the defender.

A relevant starting point for modeling debate are models of logical reasoning and human interaction. Several of these models have been implemented in the ACT-R cognitive architecture (Anderson et al., 2004). Analogical reasoning for example was modelled to solve Raven's Progressive Matrices (Ragni & Neubert, 2012), which are frequently used in IQ-tests. Objects in the cells of matrices are decomposed into different attributes, that are used to identify the rules that allow predicting the missing element of the matrix. Inference rules are implemented as ACT-R productions and the rule currently being checked is encoded in a chunk slot. Similar mechanisms could be used in a model of monastic debate.

In a different study (Ghosh, Halder, Sharma, & Verbrugge, 2015) strategies based on forward and backward induction in sequential games were investigated. A logic language was created to describe strategies and beliefs, which was then used heavily to model reasoning rules in a cognitive model. Strategies were selected based on expected payoffs. This mechanism would be more challenging to implement for monastic debate, since payoffs cannot be easily defined given that they are defined by high-level attributes such as novel insights.

A crucial ingredient for debating is theory of mind. There exist several computational models of theory of mind, which describe theory of mind as a sequence of reasoning steps with complexity dependent on the order of theory of mind, e.g. (Meijering, Taatgen, van Rijn, & Verbrugge, 2014). Successful debating is likely to involve theory of mind as well in the sense that possible moves of the opponent must be predicted to effectively trap them into a contradiction. Several strategies employed a theory of mind, including second-level theory of mind and higher. Another strategy was chosen, if the current one proved unsuccessful after a number of iterations.

A debate model necessarily requires interaction. A basis for modelling this can be found in modelling negotiation

skills (Stevens et al., 2018), where a cognitive model was created capable of using different strategies. The model made decisions based on instances of the current state of the game using ACT-R's partial matching feature. Instances were either encoded into the model's declarative memory or learned (instance-based learning). Humans who played against the cognitive model then showed improvement in their negotiation skills in terms of payoff in the Game of Nines. A simple graphical computer interface allowed interaction with the model. For a model of debate this shows that is well possible to allow interaction to train improve task-specific skills, and to let a cognitive model pursue and switch between strategies, which is important for the challenger on a experienced level.

To conclude, models of debate can be based on previous models of sequential games, logical inference, and theory of mind. On the basis of this insight we attempted to model monastic debate.

Methods

ACT-R

The cognitive model of debate was implemented in ACT-R, after previous research showed that it is well possible to model interacting and reasoning agents with this cognitive architecture (Ragni & Neubert, 2012; Ghosh et al., 2015; Meijering et al., 2014; Stevens et al., 2018). ACT-R is an excellent tool for this, because it is commonly used, well maintained, and several relevant models have been created in the past of which parts can form the basis for modelling monastic debate.

ACT-R largely consists of modules that represent executive functions of humans, e.g. visual perception or motor execution. The behaviour of the model is defined by a production system, which represents procedural memory. A single production rule can be understood as versatile if-then-statement. Model behaviour can be modulated by so-called subsymbolic features. For example, based on spreading activation or the similarity of chunks in the declarative memory, it is possible to model phenomena such as reasoning errors caused by confusing concepts or by mistaking relations between concepts.

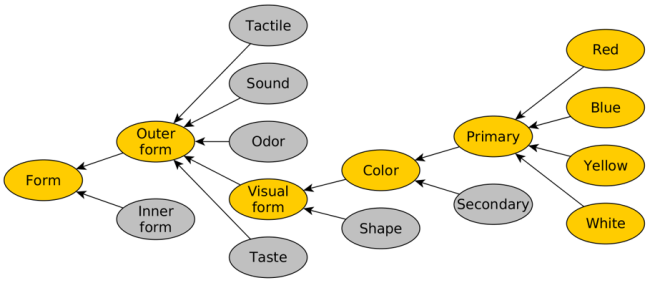


Figure 2: A sub-graph of the "colour map" (Tharpa & Tsultrim, 2012), a hierarchically organised topic of debate often used by learning debaters. Child nodes of the nodes marked in grey have been omitted for brevity.

Knowledge representation

Reasoning agents need something to reason about. Traditionally, the content of a debate is a topic from Buddhist teachings which was discussed in recent classes of the monastic curriculum. Such material is the first choice as object of debate for the model, because such traditional topics were used and explicated in real debate over thousands of years. Moreover, using the traditional debate ontology allows us to validate the model with Tibetan monks more easily (see the section on validation below).

In their first classes learning debaters usually reason about ontologies of colours and forms (Perdue, 2014), mental and physical phenomena, causation, and other topics. Each debate topic is limited to a finite set of concepts and how those concepts relate to each other, for example in *is-a*, *has-a*, *causes*, and other relations. For the cognitive model the topic of colours and forms is used as the conceptual space, which is based on *is-a* relations between concepts. To this end the knowledge tree found in the literature was replicated as knowledge base for the model.[1] An excerpt of the knowledge tree can be seen in figure 2.

Most of the listed topics found in debate tutorials that were prepared for a Western audience (Tharpa & Tsultrim, 2012) showed a hierarchical structure.[2] Hierarchically structured data has the advantage that it can be described as a tree in our model by means of graph theory. Graph theoretical approaches have been studied and applied extensively in the past, which is know-how that can be drawn from. Using graph theory as basis to represent knowledge is helpful and simplifies the conception and description of ways to manipulate a knowledge base.

While the data found in Buddhist literature can be expressed as a tree, for our formal description it is only assumed that a debater's knowledge can be represented by a directed graph.[3] For a debater d (challenger or defender) in one instance of a debate, let $G_d = (V_d, E_d, P)$ be a directed graph representing the debate-specific knowledge of d, where V_d are the nodes, E_d are the edges of the graph, and P is a single binary predicate. An example of a knowledge graph including nodes and edges can be seen in figure 2.

More specifically, $V_d = \{v_1, v_2, \ldots, v_n\}$ is a set of nodes that represents the concepts that d knows. For example, v_1 might represent the concept "colour red", v_2 the concept "primary colour", and v_3 the general concept "colour". P is a binary predicate that represents a transitive relation between concepts, for example an *is-a* relation. Transitive relation here means that if $P(v_1, v_2)$ and $P(v_2, v_3)$, then $P(v_1, v_3)$ for

[1] While learning the correct relation between concepts is a desired practice and learning new concepts in conversational agents is surely possible in principle, it is not clear whether learning new concepts is common in real debate.

[2] The reviewed literature covers topics of the first year of the monastic curriculum at the Sera Jey monastery. We do not know yet whether topics from higher years are also organised hierarchically.

[3] To perfectly represent traditional material structured in one tree, a graph needs to be acyclic and only have a single parent node.

$\{v_1, v_2, v_3\} \in V_d$. For each directed edge $(v_1, v_2) \in E_d$ (with $E_d \subseteq V_d \times V_d$) it holds that $P(v_1, v_2)$. For example, if P represents an *is-a* relation, v_1 represents the concept "colour red", v_2 represents the concept "primary colour" and $(v_1, v_2) \in E_d$ is a directed edge, then d can infer that "colour red" is a "primary colour".

In the knowledge graph concepts are represented by nodes, while in the model they are represented by ACT-R chunks and only include a slot for the name. A concept is retrieved based on its name, which may appear as a word when processing auditory input. Pervasions (directed edges in the graph) are modelled by chunks with two slots that contain those concepts. In ACT-R this is expressed as a chunk as follows in the example of the pervasion "color is a visual form":

```
(per_color_is_a_visual_form
 isa         pervasion
 pervaded    c_color
 pervader    c_visual_form)
```

The pervaded and the pervader can be likened to the antecedent and consequent in a material implication in formal logic. Depending on the specific debate statement, the model recalls a pervasion by matching either one or both concepts, as described in the next section. If the model encounters an unknown or otherwise unexpected word, the model responds that it cannot understand the input given.

Reasoning with graphs

Knowing details does not mean one knows the bigger picture. If a debater d has knowledge in form of a tree $G_d = (V_d, E_d, P)$ and d knows $e_1 = (v_a, v_b)$ and $e_2 = (v_b, v_c)$ (i.e. $\{e_1, e_2\} \subseteq V_d$), then d however does not necessarily know $e_3 = (v_a, v_c)$ (i.e. not generally $e_3 \in V_d$). However, d may deduce this new fact based on the property of transitivity of P. This then adds a new edge to V_d (i.e. $V'_d = V_d \cup \{e_3\}$). In other words, if P represents an *is-a* relation, d can apply the syllogism "All [tones of] red are primary colours; All primary colours are colours; Therefore, all [tones of] red are colours". ACT-R was used to create a model of both the defender and the challenger each that implements this form of deductive reasoning. At the moment only the defender uses this inference to gain new knowledge.

One of the statements the challenger might issue is the "three-part debate" statement. For example, if d is a challenger and P represents an *is-a* relation between concepts, then three-part debate typically follows the form "Take the subject $\langle subject \rangle$, it follows it is $\langle predicate \rangle$, because it is $\langle reason \rangle$." where the three placeholders represent concepts in V_d. When a three-part debate statement is given to the defender, deductive inference allows to to confirm or reject the statement. For example, let the defender know "Socrates is a man" and "All men are mortal". If the challenger inquires "Is Socrates a man?" the defender can immediately respond with "I accept.", since the defender possesses exactly that requested piece of knowledge. However, if the challenger asks "Is Socrates mortal?", then the defender should be able to infer the proposition in question ("Socrates is mortal.") from given knowledge.

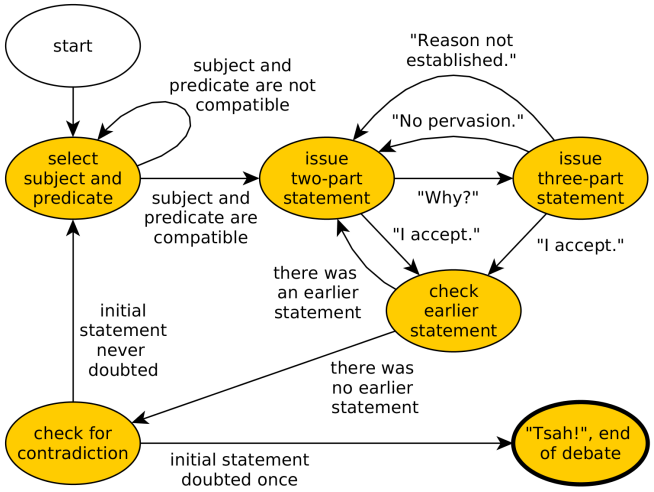


Figure 3: A visualisation of the course of debate with limited ways to respond, from the point of view of a challenger.

Based on the *negation as failure*-principle, the defender model will accept the logical proposition of the debate statement, if the proposition can be inferred deductively in one or more steps based on the knowledge the defender has. If the logical proposition cannot be inferred, it will be rejected with the words "Reason not established" or "No pervasion" (just like in real debate), depending on at which point the inference fails.

Model evaluation

Commonly ACT-R models are evaluated by comparing model and human performance with regards to reaction times, accuracy of task performance, or fMRI or EEG measurements. Refining the model until its performance matches human performance, including error during performance, is a method to validate the model.

However, monastic debate does not follow a typical task paradigm, which makes it hard to apply standard methods. Due to the complexity of debate it is difficult to measure reaction times. Capturing the accuracy of a debater is difficult, as there are no typical metrics to measure whether a debater performs well during debate. Debate measures related to accuracy that are typical for experimental tasks might be the number and type of reasoning errors, the length and duration of debate, how often the defender responds in a certain way, or how often the challenger changes strategies. However, those data first need to be collected in some way and even then comparison might still be tricky, since any two debates rarely follow the same course.

To acquire data from the model two interfaces were created for interaction with the model. A terminal-based interface allows for quick testing and automation during model creation. The other interface is runs in web browsers and offers a few conveniences. Graphical user interfaces are often

more accessible, can provide an easier way to change relevant parameters, and generally allow a better user experience for a less technology-savvy audience. This is important, as the browser-based interface is intended to be used for data acquisition and model validation in the future. A welcome side effect in the spirit of open science is that the browser-based design is easy to adapt for access via the internet[4].

Results

While the defender remains mostly reactive, the challenger needs to plan ahead what statements to issue at what time, to eventually lead the defender to accept a contradicting statement. For a better understanding of the directions a debate can take, it might help to focus not on one of the debaters specifically, but their interactions over time. The possible paths of a very limited form of debate are visualised in figure 3. The diagram can be read like a state diagram of a finite state automaton. Note that the diagram contains the essential two-part and three-part debate statements, but no other types of statements. To capture the more diverse and complex human debates fully the diagram would need to be extended.

An example interaction via the terminal-based interface between a human defender and the challenger model can be seen in figure 4. In the example, the relation between concepts is an *is-a* relation, so the two-part and three-part debate statements take the general form "Take the subject ⟨*subject*⟩, it follows it is ⟨*predicate*⟩[, because it is ⟨*reason*⟩].".

In the beginning the challenger posits a two-part debate statements until the the defender does not accept. This provides the starting point for the debate and the challenger now tries to make the defender accept the same statement. As next step the two-part debate statement will be extended to a three-part debate statement by providing a ⟨*reason*⟩. The ⟨*reason*⟩ essentially is one of the concepts represented by a node in the knowledge graph, that lies on the path between the nodes for the ⟨*subject*⟩ and the ⟨*predicate*⟩ in the two-part debate statement. If the defender disagrees that "⟨*subject*⟩ is a ⟨*reason*⟩", a "Reason not established" response follows, but if the defender disagrees that "⟨*reason*⟩ is a ⟨*predicate*⟩", a "No pervasion" response follows. In both cases the challenger then knows the two concepts for which an *is-a* relation is denied and so issues a new two-part debate statement based on those two concepts[5]. After issuing this new statement the same procedure as just described follows, so this can be considered a new step in a nested or recursive procedure.

However, the defender might reject a two-part debate statement, but not the respective three-part debate statement. This is the case, when the defender has the relations "⟨*subject*⟩ is a ⟨*reason*⟩" and "⟨*reason*⟩ is a ⟨*predicate*⟩" represented as edges in the knowledge graph. Then the defender accepts the three-part statement, which could be considered going back one step in the recursive procedure. If the defender accepts the statement that was rejected in the beginning, the challenger may declare the uncovering of a contradiction with a "Tsah!" response and the debate ends.

C:	Take the subject rough, it follows that it is form.
D:	`why`
C:	Take the subject rough, it follows that it is form, because it is outer form.
D:	`no-reason`
C:	Take the subject rough, it follows that it is outer form.
D:	`why`
C:	Take the subject rough, it follows that it is outer form, because it is tactile.
D:	`no-reason`
C:	Take the subject rough, it follows that it is tactile.
D:	`why`
C:	Take the subject rough, it follows that it is tactile, because it is derived from element.
D:	`accept`
C:	Take the subject rough, it follows that it is outer form, because it is tactile.
D:	`accept`
C:	Take the subject rough, it follows that it is form, because it is outer form.
D:	`accept`
C:	Tsah!

Figure 4: The model in action: An example debate with a challenger model (denoted *C*) and a human defender (denoted *D*). The simplified responses of the human defender are interpreted as the written out, formally correct debate statements.

In contrast to the challenger model, the defender model is simpler, yet also based on the course of debate shown in figure 3. Generally, a two-part debate statement is accepted, if the defender knows that "⟨*subject*⟩ is a ⟨*predicate*⟩". For a three-part debate statement the defender will check whether "⟨*subject*⟩ is a ⟨*reason*⟩" and "⟨*reason*⟩ is a ⟨*predicate*⟩" and react as described above. Accepting a three-part debate statement means that the model "has an insight" and learned that "⟨*subject*⟩ is a ⟨*predicate*⟩", i.e. added an edge to the knowledge graph.

This shows that the model can function in the simplified world of the first-year debate material. The model can take the role of both a challenger and a defender that evaluate two-part and three-part debate statements in the current version of the model. Reasoning is performed error-free on the basis of complete (challenger) or incomplete (defender) knowledge.

[4] At the time of writing, a recent state of the model can be found at `https://blueparrot.pythonanywhere.com/`. This address might updated in the future.

[5] From an algorithmic point of view it is not necessary to restate the previously rejected part of the three-part as two-part debate statement, as the defender will always consistently respond with "Why?" to it. However, in real debate it is generally considered good practice to be explicit, which should then also be captured by a model.

Discussion

Validation

Monastic debate involves many cognitive processes and different ways to interact with the environment, including high-level executive functions such as theory of mind. Due to the complexity of debate it seems difficult to apply validation methods that are typically used for ACT-R models, such as correlating reaction times. In addition, it appears tricky to test isolated parts of debate, as essential aspects of this integrative practise might simply get lost in a controlled lab setting. Investigating cognition "in the wild" is an objective of the scientific community of macrocognition and their methodology such as Cognitive Task Analysis (Crandall, Klein, Klein, & Hoffman, 2006) might help to investigate debate.

However, certain quantitative properties of debate could be collected and compared, such as the level of a debater's expertise, the number of agreements between debaters (van Vugt et al., 2019), or the number of reasoning errors per debate. As an additional way to collect such data and as a separate form of model validation a "Monastic Turing Test" could be performed, i.e. a Turing Test[6] that allows a human judge to rate responses from another debater. The human judge is debating with the partner via a text chat and normalised responses, but the judge does not know whether the other debater is another human or one of the challenger or defender models.

To be able to perform such a monastic Turing test, a browser-based interface was created to interact with the model. Such an interface has the additional benefit that researchers can collect normalised data in an automated way, which is helpful for further assessment of debates and comparison with model behaviour.

Before starting a debate via the interface, a human debater is able to select whether the role of the challenger should be filled by a human or by the cognitive model, and similarly for the defender. Random assignment of the judge to a model or a human is possible, which allows to realise the Monastic Turing Test. If a certain model is judged to be more human than other models after a number of iterations, then the model may follow certain strategies or make certain errors that resemble those of humans more closely.

Contrasting candidate models that differ in ACT-R model parameters or implemented strategies can be considered a "relative comparison", which does not yet validate a model (Palminteri, Wyart, & Koechlin, 2017). Hence it is imperative to evaluate single models based on their "generative performance", i.e. whether and how well a model is able to reproduce evidence, including different debate strategies. Nevertheless, we think that our model does currently not match human performance well, because it does not make errors yet.

[6]In the text the Turing Test is based on the modern, most common interpretation, not in the sense of the "imitation game" as originally proposed by Allan Turing.

Reasoning errors

Errare humanum est. Humans do not always perform optimal and one way to make the model behave more human-like is to introduce errors. In human debates however mistakes do occur frequently, especially due to memory failures or time pressure, since taking too long to respond is considered to be a weakness (Dreyfus, 2003). Two common sources of errors appeared to be an inconsistent body of knowledge and an incorrect application of rules of logical inference for debate (reasoning errors).

Errors due to an inconsistent knowledge were modelled by a manipulated knowledge graph, where edges were added or removed such that the formal definition of a knowledge graph is violated. In those cases the model then returned erroneous responses, for example the model rejected that red is a primary colour or accepted that primary colours are both colours and sounds.

A reasoning error is given, when only valid premises are given (drawn from the body of knowledge or from the ongoing debate), but the reasoner still arrives at an incorrect conclusion. In human debate this is not uncommon, especially among novice debaters. ACT-R has a set of so called subsymbolic features like spreading activation or partial matching, which will aid the implementation of such reasoning errors.

There can be many reasons for reasoning errors, for example the defender can get nervous, confused by long statements, distracted by the environment, exhausted from a long debate or simply forget earlier statements. In ACT-R a defender's confusion of concepts could be modelled, for example, by encoding the similarity between (memory) chunks and allowing the retrieval of a similar, but incorrect chunk, when listening to the words of the challenger. Integrating reasoning errors into the model however is part of future research.

Future work

While considerable progress was made in creating a computational model of debate, there are limitations to the model.

Firstly, it is possible that our assumption of hierarchically-structured knowledge is too strong to represent the majority of real-life debates. Future models should attempt to relax this restriction. In addition, dealing with more general knowledge structures opens up the possibility to reason about more complicated topics of the monastic curriculum. Topics foreign to the Tibetan tradition might then also be considered, such as study material frequently discussed in Western education systems.

As mentioned in the earlier many forms of response types exist in debate, but they are used in different frequencies and not all are equally easy to formalise. Not all debates have a challenger who requests an example or the reason for a certain statement (see table 1). However the two-part and three-part debate statements and their responses are essential and could not be taken away without losing defining aspects of debate. Extending the model by adding more ways to interact makes the debate more engaging and the model more realistic.

Experienced challengers often switch between different debate styles, which might be less defined by what kinds of statements are issued, but more how the parts are combined to lead the debate. Such styles can differ in e.g. the use of analogies, the ratio between exploration and exploitation of debate subjects, referring back to previous debate subjects, the trade-off between fast vs. accurate responses, or attempts to trick the defender. Capturing such notions formally and integrating them into the model allows the model to match human behaviour more closely.

Conclusion

In this work an ACT-R model was created that uses graph theory for flexible and extensible knowledge representation. This innovative approach captures essential parts of simple debate instances, but also a cognitive process, which both have not been described formally before.

More generally, we think this research provides a promising early stage model of an interaction in a complex real-world environment. We are confident that by looking at tasks outside the ordinary domain, we can gain insights into cognitive skills that generalise better across all human beings, not only minds trained in Western education (Henrich, Heine, & Norenzayan, 2010).

Acknowledgements

The authors wish to thank Losang Donyo, a monk from the Sera monastery, for the joint sessions of debate and advisory support; and Amir Moye, who provided valuable input and feedback during model creation.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036.

Crandall, B., Klein, G., Klein, G. A., & Hoffman, R. R. (2006). *Working minds: A practitioner's guide to cognitive task analysis*. Mit Press.

Dreyfus, G. (2003). *The sound of two hands clapping: The education of a tibetan buddhist monk*. Univ of California Press.

Ghosh, S., Halder, T., Sharma, K., & Verbrugge, R. (2015). Human strategic reasoning in dynamic games: Experiments, logics, cognitive models. In *International workshop on logic, rationality and interaction* (pp. 116–128).

Henrich, J., Heine, S. J., & Norenzayan, A. (2010). Most people are not weird. *Nature*, *466*(7302), 29–29.

Liberman, K. (2007). *Dialectical practice in tibetan philosophical culture: An ethnomethodological inquiry into formal reasoning*. Rowman & Littlefield Publishers.

Meijering, B., Taatgen, N. A., van Rijn, H., & Verbrugge, R. (2014). Modeling inference of mental states: As simple as possible, as complex as necessary. *Interaction Studies*, *15*(3), 455–477.

Palminteri, S., Wyart, V., & Koechlin, E. (2017). The importance of falsification in computational cognitive modeling. *Trends in cognitive sciences*, *21*(6), 425–433.

Perdue, D. E. (2014). *The course in buddhist reasoning and debate: An asian approach to analytical thinking drawn from indian and tibetan sources*. Shambhala Publications.

Ragni, M., & Neubert, S. (2012). Solving raven's iq-tests: an ai and cognitive modeling approach. In *Proceedings of the 20th european conference on artificial intelligence* (pp. 666–671).

Stevens, C. A., Daamen, J., Gaudrain, E., Renkema, T., Top, J. D., Cnossen, F., & Taatgen, N. A. (2018). Using cognitive agents to train negotiation skills. *Frontiers in psychology*, *9*, 154.

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological review*, *120*(3), 439.

Tharpa, T., & Tsultrim, N. S. (2012). *Debate Primer Text 1 - A dialectic debate primer and method text for Tibetan monastic debate in the Tibetan language.* Liberation Publications. Retrieved 2020-04-03, from `http://tenzintharpa.com/download-library/` (3rd ed. 2012)

van Vugt, M. K., Pollock, J., Johnson, B., Gyatso, K., Norbu, N., Lodroe, T., ... Khechok, J. e. a. (2020). Inter-brain synchronization in the practice of tibetan monastic debate. *Mindfulness*, 1–15.

van Vugt, M. K., Moye, A., Pollock, J., Johnson, B., Bonn-Miller, M. O., Gyatso, K., ... others (2019). Tibetan buddhist monastic debate: psychological and neuroscientific analysis of a reasoning-based analytical meditation practice. In *Progress in brain research* (Vol. 244, pp. 233–253). Elsevier.

Feedback Influences Syllogistic Strategy: An Analysis based on Joint Nonnegative Matrix Factorization

Nicolas Riesterer* **(riestern@cs.uni-freiburg.de)**
Daniel Brand* **(daniel.brand@cognition.uni-freiburg.de)**
Marco Ragni (ragni@cs.uni-freiburg.de)
Cognitive Computation Lab, University of Freiburg
Georges-Koehler-Allee 52, 79110 Freiburg, Germany

Abstract

Feedback for drawn inferences can lead to an adaption of future responses and underlying cognitive mechanisms. This article presents a reanalysis of recent hypothesis-driven experiments in syllogistic reasoning in which participants were presented with different feedback conditions (no feedback, 1s, 10s). We extend the original analysis, which only focused on no feedback vs. 1s feedback, by including the additional 10s condition. For our analysis, we rely on the data-driven theory- and hypothesis-agnostic Joint Nonnegative Matrix Factorization which allows us to contrast datasets based on the extraction of response patterns reflecting common and distinct response behavior. Our results support the previous claims that feedback does not generally boost logical reasoning ability but reduces the influence of biases against the response indicating that nothing logically follows from the premises.

Keywords: syllogistic reasoning; feedback; joint nonnegative matrix factorization; nvc bias

Introduction

It is a well-established fact that individuals do not only differ with respect to the strategies they employ but are also capable to adapting them to the problems they are confronted with (e.g., Bucciarelli & Johnson-Laird, 1999; Roberts et al., 2001). One consequence of this manifests, for instance, in terms of the effects different instructions may have on participants' performances (Dickstein, 1975).

Since the general goal of investigating reasoning is to understand the processes underlying human inference, exploring these adaption capabilities is an important paradigm of research. One approach for this goal is via feedback. A recent study (Dames et al., in press) investigated the effects of feedback in the domain of human syllogistic reasoning, which is concerned with inferences based on quantified relations (e.g., Khemlani & Johnson-Laird, 2012). In the study, participants were presented with feedback about the correctness of their conclusions after each task. The results suggest that feedback helps to boost logical correctness of responses and leads to post-error adaption effects with respect to reaction times. However, the authors also note that large parts of the improved correctness could be attributed to a substantial increase of the response "No Valid Conclusion" indicating that no quantified conclusion can logically be inferred from the premises. This casts doubt on the possible interpretation that feedback benefits logical thinking in its literal meaning.

The study provides insight into the impact of feedback on syllogistic reasoning ability on a statistical level by relying on a hypothesis-driven analysis. However, in doing so, the authors essentially apply a hypothesis-based filter to their data which could lead to additional results being left in the dark. In particular, the question remains if the reported results conclusively reflect the effects of feedback or if further influence on syllogistic response behavior remains. Investigating precisely this influence on the level of response patterns is crucial for cognitive modeling, because gaining insight into how manipulations affect human behavior on the response level could provide the information necessary to develop improved models, both on the level of predictive accuracy and explainability.

The goal of this article is therefore to obtain insight into the effects of feedback on the level of response patterns in the domain of syllogistic reasoning. To adopt a data-analytic perspective that is unbiased with respect to theoretical assumptions about syllogistic reasoning ability, we rely on an analysis using *Joint Nonnegative Matrix Factorization* (JNMF, Kim et al., 2015), a general approach for contrasting datasets that was originally introduced in the field of information systems but has since been transferred to the domain of reasoning (Brand et al., in press). By simultaneously solving a matrix factorization problem for two data matrices, JNMF allows to directly extract patterns which are common or distinct to the two input datasets. As such, the results of JNMF, in terms of interpretability and expressiveness, go beyond what single-dataset factorization methods can offer (Kim et al., 2015).

The remainder of this article is structured into four parts. First, we introduce the relevant background literature about syllogistic reasoning and the application of JNMF. Second, we present the methodology of our analysis. Third, we present the results, i.e., the extracted response patterns and interpret them in terms of the effects of feedback. A discussion of the implications of our analysis concludes the article.

Related Work

Syllogistic reasoning is one of the central domains investigated in research of human deductive reasoning ability (e.g., Johnson-Laird & Byrne, 1991). A syllogism consists of two premises which specify quantified relationships (using quantifiers *All*, *Some*, *No*, *Some ... not*) between three categorical terms (e.g., A, B, C):

No A are B
Some B are not C

What, if anything, follows?

The goal in syllogistic reasoning is to use the middle term, B, which occurs in both premises to infer information about the remaining two terms, A and C (end terms). In total, the syllogistic reasoning domain consists of 64 distinct problems which are obtained from the 16 possible combinations of premise quantifiers

*Both authors contributed equally to this manuscript.

and the four possible arrangement of terms in the premise (the so-called *figures*; see Khemlani & Johnson-Laird, 2012). Each syllogistic problem has nine possible propositional conclusions, eight of which relate the end terms (in either direction) using one of the four quantifiers, and the conclusion "No Valid Conclusion" (NVC) to indicate that, in accordance to first-order logic, no quantified conclusion follows from the premises.

A core result of syllogistic reasoning research is that human inferences deviate substantially from what classical logic would predict (e.g., Khemlani & Johnson-Laird, 2012). Because of this, research in the domain has focused in large parts on the development of high-level cognitive theories and corresponding model implementations (for a review, see Khemlani & Johnson-Laird, 2012). Recently, the interest in the effects of interindividual differences has been rekindled by studies focusing on different subgroups of reasoners (Khemlani & Johnson-Laird, 2016) and analyses of the shortcomings of current models when subjected to individual response data (Riesterer et al., 2019) being published.

Analyzing individual human reasoning behavior, it could be shown that human performance in syllogistic reasoning tasks is far from robust. If participants respond to syllogisms in two sessions one week apart from each other, logical correctness improves even though participants are not provided with feedback to their responses (Johnson-Laird & Steedman, 1978; Ragni et al., 2018). Similarly, within the sequence of 64 syllogistic problems, it can be observed that the likelihood of giving NVC conclusions rises as a function of presented problems, causing logical correctness to rise (Ragni et al., 2019).

To investigate the susceptibility to changes in syllogistic response behavior, a recent study adopted a paradigm in which reasoners were presented with immediate feedback for different durations indicating the correctness of their responses (Dames et al., in press). Analyzing the resulting response data, it could be shown that participants who were not provided with feedback tended to give less logically correct conclusions than participants in the feedback group. Additionally, it could be shown that feedback induced post-error adaption effects causing reaction times to slow down. The results suggest that participants are capable of adapting their response behavior in light of feedback. However, the authors also note that participants who received feedback responded substantially more often with NVC conclusions. They argue that this effect could be due to a bias or aversion against NVC responses which is overcome by providing feedback. This is a hypothesis that has been discussed, albeit inconclusively, in the literature of syllogistic reasoning before (e.g., Revlis, 1975; Roberts et al., 2001).

In the following analyses, we want to push the statistical work of Dames et al. (in press) one step further to obtain results on the level of behavioral patterns which could lead to information useful for improving models of human syllogistic reasoning. To this end, we base our analysis on contrasting.

Joint Nonnegative Matrix Factorization

Contrasting refers to the problem of finding iconic distinction factors that best describe the differences between datasets. Trivially, computing differences is one way of performing contrasting. However, given structurally rich data that are potentially noisy, the results of trivial contrasting is lacking with respect to their potential for interpretation. More sophisticated contrasting is based on the results of factor analyses. For example, by performing *Principal Component Analysis* (PCA; e.g., Murphy, 2012), the dimensionality of data can effectively be reduced to a smaller number of k latent features which can then serve as the basis for dataset comparison. However, for two independent applications of PCA such as for two separate datasets in a contrasting scenario, there is no guarantee that the resulting factorizations are related and support comparison. Especially if differences between datasets are expected to be small, it is important to factor out the commonalities in order to expose the crucial distinctions.

Motivated by this problem, work in the field of information systems has developed *Joint Nonnegative Matrix Factorization* (JNMF; Kim et al., 2015), an approach for contrasting two datasets via matrix decomposition. To achieve this, JNMF extends regular *Nonnegative Matrix Factorization* (NMF; see Pauca et al., 2004) by simultaneously searching for factors representing commonalities and distinctions between both datasets.

Formally, NMF is the problem of finding a decomposition of a single data matrix $X \in \mathbb{R}^{m \times n}$ where m is the dimensionality of the data and n denotes the number of objects in the dataset into a basis matrix $W \in \mathbb{R}^{m \times k}$ and coefficient matrix $H \in \mathbb{R}^{n \times k}$ where $k < \min\{m,n\}$ is the number of patterns, or factors, to decompose the data into, such that

$$X \approx WH^T \tag{1}$$

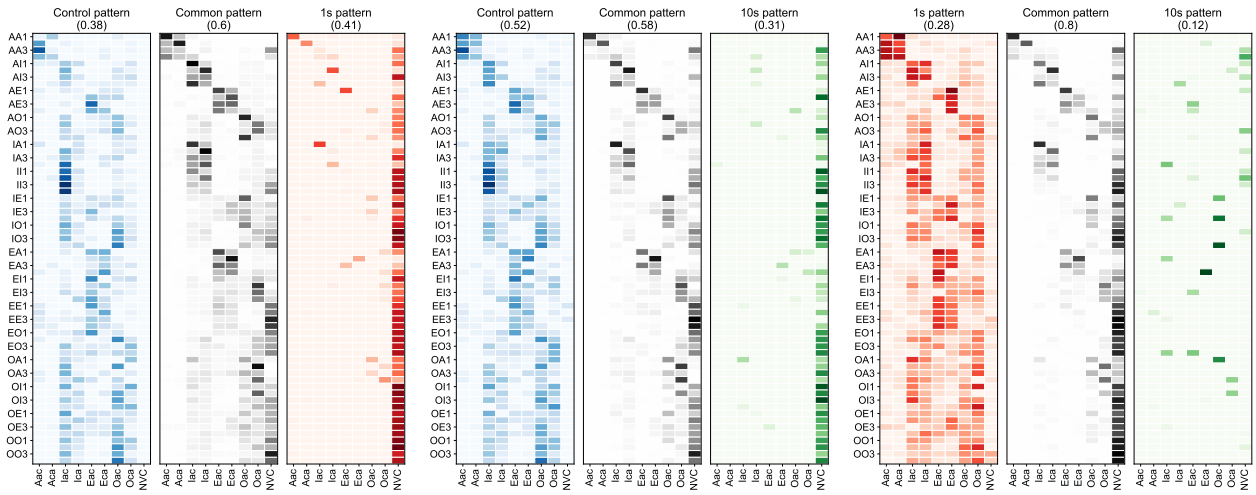
JNMF refers to the problem of finding a decomposition of two data matrices $X_1 \in \mathbb{R}^{m \times n_1}$ and $X_1 \in \mathbb{R}^{m \times n_2}$ into a basis matrix $W_i \in \mathbb{R}^{m \times k}$ and a coefficient matrix $H_i \in \mathbb{R}^{n_i \times k}$ for $i = 1,2$ (Kim et al., 2015). Crucially, $k = k_c + k_d$ refers to the total number of patterns consisting of k_c common and k_d distinct patterns which means that $W_i = [W_{i,c}, W_{i,d}]$ is composed of columns referring to k_c common ($W_{i,c}$) and k_d distinct patterns ($W_{i,d}$). The goal of JNMF is to find matrices W_1, H_1 and W_2, H_2 solving Equation (1) for both data matrices X_1 and X_2 under constraints regularizing the identified W and H matrices to ensure that the distances between common and distinct patterns are minimized and maximized, respectively (Kim et al., 2015).

An important advantage of NMF (or JNMF for that matter) for its application in the context of cognitive science is its focus on nonnegativity. Since data obtained in behavioral experimentation is usually nonnegative as well (response choices, reaction times, etc.), NMF operates directly and natively on the expected range of values which allows for better interpretability of the results (Pauca et al., 2004).

Stemming originally from the field of information systems, JNMF has recently been applied successfully to syllogistic reasoning data (Brand et al., in press). In this analysis, the authors investigated the influence of personality factors on syllogistic reasoning behavior. Using JNMF, they managed to extract the patterns distinctly representing the response behavior of participants with varying scores on personality traits. In the following analyses we apply this method to feedback data.

Method

Our goal is to investigate the effects of feedback on the responses given to syllogistic reasoning problems. To this end, we employ



(a) Control vs. 1s. (b) Control vs. 10s. (c) 1s vs. 10s.

Figure 1: Typical response patterns obtained from JNMF application and arranged as matrices of 64 rows (syllogistic problems) and 9 columns (possible conclusions). Common patterns represent the average common pattern from W_1 and W_2. The values in parentheses denote the relative importances of the patterns for the reconstruction of the dataset. These values are derived from the respective columns of the H-matrices.

JNMF in order to decompose the data into common and distinct patterns which can then be interpreted directly. Relying on JNMF allows us to adopt a theory- and hypothesis-agnostic perspective which, in turn, allows us to obtain comprehensive results which could potentially go beyond the findings of Dames et al. (in press).

Dataset

For our analysis, we rely on a dataset that was published recently as part of a study of feedback effects in syllogistic reasoning (Dames et al., in press). The focus of the study was to investigate the influence of feedback on participants' propensities to give logically valid conclusions (Dames et al., in press, Experiment 1) and on their reaction times (Dames et al., in press, Experiment 2). To this end, the authors conducted a series of experiments via Amazon Mechanical Turk in which participants were instructed to give conclusions to all 64 syllogistic problems. In total, the dataset comprises three conditions: a control group which received no feedback ($n = 39$), a group which was presented with short feedback (1s, $n = 102$), and a group which was presented with extended feedback (10s, $n = 29$). In their analysis, the authors focused on the control and 1s conditions. The effects of extended feedback have not been published up till now.

Data Preparation

To make the data accessible to JNMF analysis, we first transform the response data for each condition into matrix representations. This is achieved by onehot-encoding individual responses as zero-vectors of dimensionality 9 in which a single 1 indicates the corresponding response. As examples, for the syllogistic response "All A are C" this leads to the onehot-encoded vector $(1,0,0,0,0,0,0,0,0)$, and for "No Valid Conclusion" to $(0,0,0,0,0,0,0,0,1)$. Concatenating the 64 onehot-encoded responses of a single participant results in a 576-dimensional vector which reflects their individual response pattern, or *reasoner profile* (Riesterer et al., 2019; Brand et al., in press). By arranging all reasoner profiles as column vectors in a matrix, we obtain $m \times n$ matrices, where m is the number of features, i.e., the dimensionality of our reasoner profiles ($m = 576$) and n denotes the number of participants in our datasets ($n_1 = 39$, $n_2 = 102$, and $n_3 = 29$ for no feedback, short feedback, and extended feedback, respectively). The following analyses are computed directly on these data matrices. The data and analysis scripts are publicly available on GitHub[1].

Results

Our analyses are based on JNMF application for pairwise contrasting of the three datasets. For each pair of datasets, JNMF produces W and H matrices containing the common and distinct response patterns and their weightings for reconstructing each individual from the input data, respectively.

Pattern Analysis

The general results of the JNMF application are depicted in Figure 1. The heatmaps visualize the patterns extracted from the W-matrices in pairwise contrasting applications for the three feedback conditions: control vs. 1s (Figure 1a), control vs. 10s (Figure 1b), and 1s vs. 10s (Figure 1c). In each subplot, the patterns are presented as heatmaps with distinct patterns located left and right, and the common pattern being depicted as the mean of the common vectors from W_1 and W_2 in the middle. The shading of cells indicates the weighting of individual responses for the patterns in accordance to the values in W. The values in parentheses next to the titles denote the importances of the patterns for the reconstruction of the original dataset which were calculated from the sum of the

[1] https://github.com/nriesterer/iccm-nmffeedback

Table 1: Proportion to reassignment of individuals from source groups (rows) to target groups (columns) based on response pattern similarities. Ties were resolved by ignoring participants which causes percentages to not sum up to 1.

	control	1s	10s
control		77%	15%
1s	32%		59%
10s	14%	83%	

respective columns in the corresponding H matrix normalized by the total sum of the H matrix. These values show that, overall, commonalities are more important than distinctions which is to be expected in a complex task such as syllogistic reasoning.

Both, the contrasting between control and 1s (Figure 1a), and between control and 10s (Figure 1b) show that the distinct differences between the datasets manifest in terms of the dominance of NVC responses in the feedback groups. Contrasting both feedback groups (Figure 1c), a different picture emerges. Here, the 10s group yields a pattern that looks sparse and scattered in comparison to the other patterns. When taking the importance of the pattern into consideration, it becomes apparent that the JNMF assigned only low importance to the 10s pattern suggesting that the common pattern suffices to reconstruct the original data.

The results suggest that the 1s group reflects a mixture between the control and 10s group. Contrasted with control, it appears similar to the 10s pattern. However, when contrasted with 10s, it appears similar to the control pattern. Evidence supporting this assumption can be obtained from the similarities between the response behaviors of individuals which is represented in Table 1. For a participant from one of the conditions (denoted as source group), we checked which of the other conditions (target groups) contained the individual most similar to them. The values therefore represent proportions to which individuals from one group prefer another in terms of similar response behavior (ties were resolved by ignoring the participants which is why values do not sum to 100%). The results illustrate the special role of the 1s group. While both control and 10s clearly favor 1s over another, 1s is much more evenly distributed which indicates that it consists both of participants showing feedback behavior and participants who are still unaffected by it. In case of 10s, the proportion of individuals performing similar to controls is reduced substantially.

Put together, the contrastings provide evidence for the NVC aversion hypothesis (e.g., Revlis, 1975). Contrasted against control, the distinct patterns of both, 1s and 10s focus chiefly on the NVC response. However, when comparing the feedback patterns resulting from contrasting with control, it appears as if the NVC dominance is stronger for 1s than for 10s which could hint at a time-dependent effect of feedback. As previously concluded by Dames et al. (in press), given short (1s) feedback, a lot of participants seem to quickly grasp the importance of NVC. Combined with the evidence obtained from the extended (10s) condition, it appears as if the lack of time to reflect the meaning of NVC results in participants overestimating the frequency of invalid syllogisms. There still remain some individuals, though, who are unaffected by feedback resulting in 1s representing a mixture between reasoning patterns related to the control and 10s groups. In case of extended feedback, participants are given time to reflect their use of NVC which may lead to a more deliberative and careful reliance on this response affecting most of the individuals in the data. Consequently, when contrasting between both feedback conditions, due to the similarities of their NVC reliance, JNMF mainly uses the common pattern to capture the feedback reasoning behavior (including NVC) and uses the distinct patterns to capture residual responses.

Prediction Analysis

While the previous analysis illustrated the general structure and properties of the response patterns for the different conditions of feedback, their quality still remains obscure. To evaluate this, we now interpret the obtained patterns as predictive models and subject them to an analysis in which the accuracy of JNMF patterns in accounting for individual reasoners' responses is assessed (for the predictive task, see Riesterer et al., 2019). In this analysis, we expect patterns to perform best with respect to predicting responses on their respective conditions (i.e., the control pattern on the control dataset and so on). Simultaneously, we expect patterns to perform in proportion to their importances, i.e., to the number of individuals for which the pattern is crucial when reconstructing the response data.

Figure 2 depicts the results of this analysis. Datasets on which the patterns are evaluated are depicted on the x-axis while the y-axis denotes the predictive accuracies they achieve. Each line reflects a pattern with colors indicating their respective conditions (grey, blue, red, and green representing common, control, 1s, and 10s, respectively).

On a high-level, comparing the patterns shows that, overall, the common patterns achieve the highest predictive accuracies which is in line with the findings above. Since the common patterns are most important for reconstructing the data, the distinct patterns are not expected to be good predictors of reasoning behavior on their own, while the commonalities reflecting general reasoning behavior are. Because of this, the quality of the distinct patterns should not be assessed based on absolute accuracy values but based on their ability to provide suitable predictors for their respective data conditions.

A prime example for this is the trend of the blue lines corresponding to the no-feedback control patterns. The fact that accuracy drops substantially on the feedback data indicates that the patterns are highly descriptive for the control data only and bear little meaning for the feedback groups. Additionally, the fact that control's accuracies on the feedback data is similar for 1s and 10s suggests that the differences between the two are minor. Considering the feedback patterns obtained from contrasting with control (dark red for 1s, dark green for 10s), a different picture emerges. Here, as expected, the patterns perform much better in accounting for the feedback data than for the control data. Put together, the blue, dark red, and dark green lines illustrate the clear distinction between the control and feedback groups. Distinct patterns obtained for either condition are suitable predictors for their own data but offer severely limited applicability to the other. Additionally, the high similarity between accuracies on both feedback groups suggests that JNMF application found patterns distinct to general feedback behavior regardless of the underlying feedback duration. The lines also show that 10s elicits the most consistent feedback behavior. While 10s patterns as

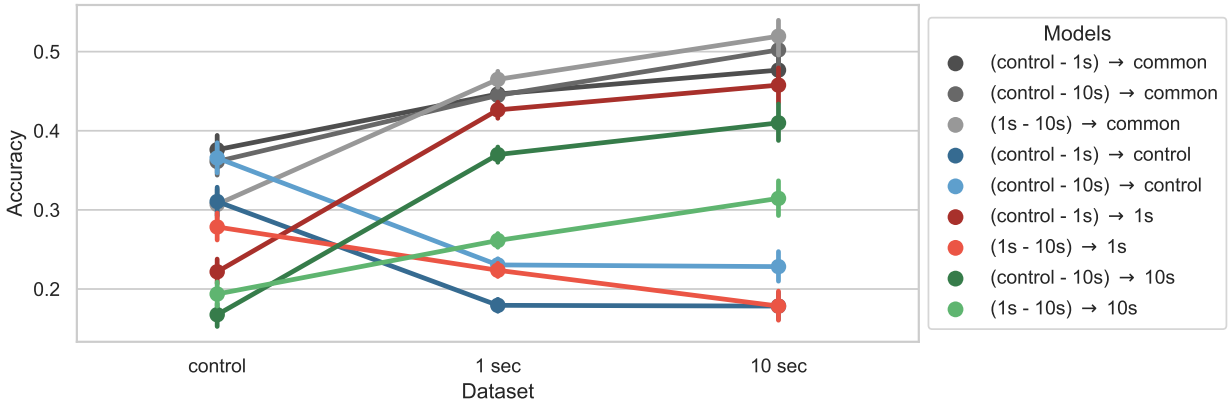


Figure 2: Accuracies of the response patterns resulting from the different contrastings (e.g., "control - 1s") interpreted as predictive models. Error bars denote 95% confidence intervals.

expected always perform best on the 10s data, this is also the case for 1s (dark red). This indicates that contrasting control with 1s does not only yield particular feedback patterns accounting for 1s, but general feedback patterns which perform even better on the 10s data.

Considering the contrasting of both feedback conditions (light red for 1s, light green for 10s), peculiar patterns emerge. The pattern for 10s (light green) performs best on the 10s data and drops substantially for the other groups thereby indicating that it captures distinct properties of the extended 10s feedback group (despite its scattered appearance in Figure 1c). The pattern for 1s (light red), however, fails to capture feedback behavior scoring higher on control data than on feedback data. Again, this indicates that 1s represents a mixed pattern. When compared to control, it clearly reflects feedback behavior. However, compared to extended feedback its distinct patterns correspond more to the control group.

The prediction analysis supports the interpretation of the results so far. The observations can be explained by assuming a time-dependent influence of feedback on reasoning behavior. Given naive reasoners, short feedback allows them to acknowledge the importance of the NVC response causing big parts of them to radically adapt their behavior in its favor (see Figure 1a) while leaving the behavior of others unchanged. Extending feedback increases the effects with diminishing returns. It appears as if the effects of extended feedback manifest in terms of a more differentiated or more deliberative use of NVC, which, at its core, is only a slight deviation from the distinct effects observable in the short feedback group. As a result, contrasting the feedback conditions leads to an overestimation of their respective differences causing the 1s pattern to be pushed towards the naive control state of reasoning behavior and the 10s pattern to focus on the very few distinct differences extended feedback results in.

Comparison of Task Performances

In a final analysis, we investigated the congruency of participant responses with formal logic in order to gain insight into whether the effects of feedback exclusively affect NVC responses as the patterns might suggest at a first glance (Figure 1) or if the effects manifest in terms of general logical correctness.

Table 2: Investigation in terms of logical correctness. The values refer to average proportions of logical correct responses and their corresponding standard errors.

Condition	Total	Valid Syllogisms	Invalid Syllogisms
Control	(33±3)%	(49±2)%	(20±4)%
1s	(46±2)%	(44±1)%	(47±2)%
10s	(50±4)%	(48±4)%	(52±6)%

The results of this analysis are summarized in Table 2. The total correctness shows that logical performance increases with prolonged feedback. When considering invalid syllogisms, i.e., the problems which do not have a propositional conclusion, the dominance of NVC responses in the feedback condition become apparent. For the valid syllogisms, however, the changes are more complex. In line with our interpretation, participants seem to overestimate the relevance of the NVC response when presented with short feedback (1s) which results in a decrease in logical correctness. With extended feedback (10s), participants seem to be able to handle NVC responses better resulting in a performance on valid syllogisms that is similar to the control condition.

In conclusion, we agree with Dames et al. (in press) that while feedback appears to boost logical correctness at first glance, it is highly unlikely that this is due to an improvement of logical reasoning ability. Additionally, based on our pattern analysis, we conclude that the observed effects can be attributed solely to a different handling of NVC responses.

General Discussion

We investigated the effects of feedback on human reasoning behavior in the domain of syllogistic reasoning using Nonnegative Matrix Factorization (JNMF; Kim et al., 2015). We were able to replicate the findings of Dames et al. (in press) who showed that feedback improves logical correctness of participants' responses mostly due to an increase in No Valid Conclusion (NVC) response frequency

on invalid syllogisms. By relying on a data-driven, theory- and hypothesis-agnostic approach and by including an extended feedback condition (10s; the authors of the original study based their analyses on the 1s condition alone), we pushed the analysis of feedback in terms of its influence on response patterns one step further.

Our results suggest that the impact of feedback depends on the duration of its presentation. Short feedback (1s) does not allow participants to properly reflect about their reasoning strategies. Instead, as Dames et al. (in press) already suggested, it only teaches them the relative importance of the NVC response which is logically correct in 37 of the 64 syllogisms (58%) and a response for which the existence of biases causing participants to reject it have frequently been assumed (e.g., Revlis, 1975). Strong evidence for this claim is found by considering the proportion of logically correct conclusions which overall increases for the short feedback condition but decreases on valid syllogisms for which NVC is an incorrect conclusion. If feedback is extended (10s), an increase of NVC responses is still the dominating distinction when compared to the control condition receiving no feedback. The logical correctness of responses to valid syllogisms remains similar, however. This suggests that the extended feedback duration allows participants to properly reflect over their reasoning strategies. As a result, the effects of increased logical correctness are in terms of invalid syllogisms only. Since performance on valid syllogisms is not affected positively by feedback (at least not significantly), in similar spirit to Dames et al. (in press), we conclude that feedback does not necessarily help to boost logical thinking in general. A more likely explanation, especially considering the differences between 1s and 10s, is, that at first feedback helps to combat an NVC aversion bias (e.g., Roberts et al., 2001) which leads to an overestimation of the relevance of NVC. Given more time to reflect, a more deliberate NVC handling with overall improved logical correctness is adopted by most reasoners.

For the general field of cognitive modeling, our findings bear relevance for two reasons. First, it is crucial to keep in mind that effects such as the lacking performance of reasoners or the corresponding increase for the feedback condition are not necessarily explained by fundamental cognitive processes. As can be shown here, rejections of particular response options must be considered by experimenters and modelers alike to ensure the proper interpretation of data and the proper development of corresponding models. Second, our analysis which was devoid of theoretical assumptions about potential inferential processes allowed us to draw comprehensive and unbiased conclusions about the available data. We strongly believe that maintaining a balance between theoretical and theory-agnostic exploration of cognition is key to ensuring a steady and uninterrupted progression of the field.

Acknowledgements

This paper was supported by DFG grants RA 1934/2-1, RA 1934/3-1 and RA 1934/4-1 to MR.

References

Brand, D., Riesterer, N., Dames, H., & Ragni, M. (in press). Analyzing the differences in human reasoning via joint nonnegative matrix factorization. In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society.*

Bucciarelli, M., & Johnson-Laird, P. (1999). Strategies in syllogistic reasoning. *Cognitive Science*, *23*(3), 247–303.

Dames, H., Schiebel, C., & Ragni, M. (in press). The role of feedback and post-error adaptations in reasoning. In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society.*

Dickstein, L. S. (1975). Effects of instructions and premise order on errors in syllogistic reasoning. *Journal of Experimental Psychology: Human Learning & Memory*, *1*(4), 376–384.

Johnson-Laird, P. N., & Byrne, R. M. J. (1991). *Deduction*. Hillsdale, NJ: Erlbaum.

Johnson-Laird, P. N., & Steedman, M. (1978). The psychology of syllogisms. *Cognitive Psychology*, *10*(1), 64–99.

Khemlani, S., & Johnson-Laird, P. N. (2012). Theories of the syllogism: A meta-analysis. *Psychological Bulletin*, *138*(3), 427–457.

Khemlani, S., & Johnson-Laird, P. N. (2016). How people differ in syllogistic reasoning. In A. Papafragou, D. Grodner, D. Mirman, & J. C. Trueswell (Eds.), *Proceedings of the 38th annual conference of the cognitive science society* (pp. 2165–2170). Austin, TX: Cognitive Science Society.

Kim, H., Choo, J., Kim, J., Reddy, C. K., & Park, H. (2015). Simultaneous discovery of common and discriminative topics via joint nonnegative matrix factorization. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining - KDD '15.* ACM Press.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. Cambridge, MA: MIT Press.

Pauca, V. P., Shahnaz, F., Berry, M. W., & Plemmons, R. J. (2004). Text mining using non-negative matrix factorizations. In M. W. Berry, U. Dayal, C. Kamath, & D. Skillicorn (Eds.), *Proceedings of the 2004 SIAM International Conference on Data Mining* (pp. 452–456). Philadelphia, PA: Society for Industrial and Applied Mathematics.

Ragni, M., Dames, H., Brand, D., & Riesterer, N. (2019). When does a reasoner respond: Nothing follows? In A. K. Goel, C. M. Seifert, & C. Freksa (Eds.), *Proceedings of the 41st Annual Conference of the Cognitive Science Society* (pp. 2640–2646). Montreal, QB: Cognitive Science Society.

Ragni, M., Riesterer, N., Khemlani, S., & Johnson-Laird, P. (2018). Individuals become more logical without feedback. In T. Rogers, M. Rau, J. Zhu, & C. Kalish (Eds.), *Proceedings of the 40th Annual Conference of the Cognitive Science Society* (pp. 1584–1589). Austin, TX: Cognitive Science Society.

Revlis, R. (1975). Syllogistic reasoning: Logical decisions from a complex data base. In R. Falmagne (Ed.), *Reasoning: Representation and process.* Hillsdale, NJ: Erlbaum.

Riesterer, N., Brand, D., & Ragni, M. (2019). Predictive modeling of individual human cognition: Upper bounds and a new perspective on performance. In T. Stewart (Ed.), *Proceedings of the 17th International Conference on Cognitive Modeling* (pp. 178–183). Waterloo, Canada: University of Waterloo.

Roberts, M. J., Newstead, S. E., & Griggs, R. A. (2001). Quantifier interpretation and syllogistic reasoning. *Thinking & Reasoning*, *7*(2), 173–204.

The Effect of Task Fidelity on Learning Curves

Frank E. Ritter (frank.ritter@psu.edu)
College of IST, Penn State
University Park, PA

Ashley F. McDermott (amcdermott@cra.com)
Charles River Analytics
Cambridge, MA

Abstract

What is the effect of level of simulation fidelity on learning and then on performance in the target task? We consider an example of an electronic maintenance training system with two levels of fidelity: a high fidelity (HiFi) simulation that basically takes as much time as the real-world task and a low fidelity (LoFi) simulation with minimal delays and many actions removed or reduced in fidelity and time. The LoFi simulation initially takes about one quarter of the time, and thus starts out providing about four times as many practice trials in a given time period. The time to perform the task modifies the learning curves for each system. The LoFi curve has a lower intercept and a steeper slope. For a small number of practice trials, this makes a significant difference. For longer time periods, the differences between low and high fidelity get smaller. Learners that move from low to high appear to not be adversely affected. We note factors that could influence this transfer (i.e., subtasks included in each simulation), and how this approach could be extended.

Keywords: fidelity, training systems, learning curves

Acknowledgements: Jake Graham, Jong Kim, Jacob Oury, Fred Ryans, Martin Yeh, and two anonymous reviewers provided useful comments. This work was supported by ONR, N00014-18-C-7015 and N00014-15-1-2275.

Introduction

What is the effect of varying the level of simulation fidelity on learning in that simulation and on the more complete learning situation? What happens to learning when a learner practices in a simpler simulation and then moves to a more realistic or higher fidelity simulation? In this paper we explore how task fidelity affects how fast a task is learned in an example task, and analyze what this means through analyses using learning curves.

We consider these questions by using an example of a maintenance training system with two levels of fidelity: (a) a simple system with minimal delays and with many actions removed or reduced in fidelity and time, and (b) a full fidelity simulation that basically takes as much time as the real world task. The higher fidelity simulations take longer to perform a more complete task including all sub-tasks. The low fidelity simulation starts out taking about one quarter of the time to complete, and thus starts out getting about four times as many practice trials in a given time period.

The task complexity in the systems influences the time to perform the task, and this in turn modifies the two learning curves, both in the intercept and in the learning rate. We will show that for a small number of practice trials, this difference in trial time makes a significant difference in the curves. For longer time periods, the differences between low and high fidelity get smaller. The amount of training the tasks being trained will receive will thus influence the choice of fidelity as well.

After briefly reviewing the effect of training system fidelity we introduce a maintenance task we have developed to study learning and retention. We will use a simple model based on ACT-R and Soar of how the task is performed and learned. Based on the learning curves we are able to draw some new conclusions about the effect of fidelity on the effectiveness of training, notably that using lower fidelity training situations help most where there is only modest time to practice, and that if there is extensive time to practice full fidelity has nearly the same outcome (but perhaps not the same costs or risks) as does starting with a simple simulation and moving to the complex simulation.

Literature Review of Fidelity

There is a long-standing debate of the effects of fidelity on training with simulators. The early research on fidelity was based on the natural assumption that higher fidelity would necessarily lead to better learning, since the simulation would more closely resemble the actual system (e.g., Allen, Hays, & Buffardi, 1986; Miller, 1954; Noble, 2002). However, much of the research supporting this notion was conducted from the 1950s to 1980s, so it had a low ceiling for how representative high simulation fidelity could be at the time. There is also a body of research showing that higher fidelity is not always desirable to maximize learning (e.g., Dahlstrom, Dekker, van Winsen, & Nyce, 2009; Havinghurst, Fields, & Fields, 2003; Lesgold, Lajoie, Bunzon, & Eggan, 1992; Swezey, Perez, & Allen, 1991).

Delving into this literature quickly leads into the question of what fidelity actually means. The most common distinction is surface or physical fidelity versus operational or task fidelity (Allen et al., 1986; Liu, Macchiarella, & Vincenzi, 2009). Within physical fidelity there are still many dimensions, including visual clutter, visual layout, auditory fidelity, and haptic fidelity. All of these dimensions have the potential to affect both speed of learning and degree of transfer to the real task. Some of these dimensions, however, are not relevant to the task being taught. To properly learn a task, the simulation should have reasonably high

fidelity on the task-relevant dimensions (Prophet & Boyd, 1970; Thorndike & Woodworth, 1901), but the irrelevant dimensions should be kept at a lower fidelity to minimize distraction from the task (Alessi, 1988).

An additional factor that affects task time and transfer of learning is the experience level of the learner (Alessi & Trollip, 1991). The experience of the learner will affect the cognitive load associated with higher fidelities and the dimensions of fidelity that could be considered task relevant (Alessi, 1988). For example, an expert who is used to using the actual interface but is doing additional training will likely experience less cognitive load with a nearly full fidelity simulation than a novice learning about the interface for the first time. Additionally, due to their experience, experts may find not having the appropriate haptic or audio cues or incorrect timings in the simulator to be a distraction to learning, while including these details would be distracting for a novice. Similarly, the age of the learner can affect what sorts of interfaces are easily usable. A low fidelity simulation could introduce interactions that are natural for younger adults but novel or slower for older adults (John & Jastrzembski, 2010).

The question of when higher fidelity is better for learning continues to be debated because it is not clear why or when lower fidelity simulations provide the most advantage. As we have discussed, experience of the learner and cognitive load are considered to be two important contributing factors, as is the type of task. In this paper we propose an additional factor, the number of repetitions of the task (or subtask) that a learner is able to complete while training.

A Simple Task Model of Learning and Fidelity

To examine the effect of fidelity on learning we use an example simulation, the Ben-Franklin (BF) Radar (Ritter, Tehranchi, Brener, & Wang, 2019). Figure 1 shows a schematic for the BF Radar system included to show its relative complexity, not its details. The system has 35 replaceable components that can have faults, and 15 switches and a power supply that cannot have faults. The system is based on the Klingon Laser Bank task (Friedrich & Ritter, 2020; Kieras & Bovair, 1984; Ritter & Bibby, 2008) and on a functional radar system (Charvat, 2011).

The schematic shows five subsystems. The subsystems vary in their complexity and connectivity within them and across subsystems. The blue lines in Figure 1 are power connections; the red lines are information; the purple lines are both. The schematic also identifies certain components that have their status displayed on the front panel of the BF Radar.

There are several tasks that can be performed with the BF radar system. Users can turn it on; users can correctly adjust switches so that it works; users can find a single fault and replace it; and users can find and replace multiple faults. The task that we will use to examine the effect of fidelity on learning is to find a single broken component, a fault. Single broken faults create a unique light configuration and are always solvable.

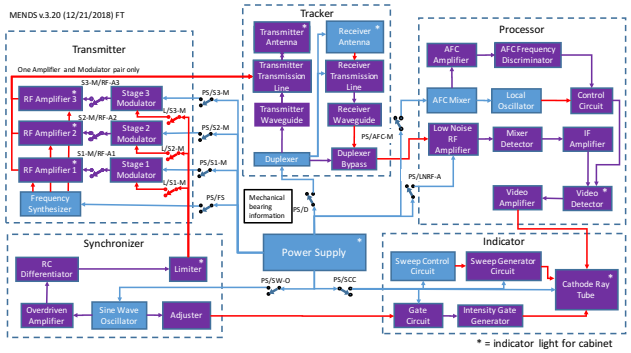


Figure 1. Schematic of the Ben-Franklin Radar simulation.

The task was created to support troubleshooting within the confines of a study, and to be more complex than the Klingon Laser Bank task, but not so complex that it would take more than an hour to learn. This system can be and has been realized in several ways with different levels of assumed fidelity.

Task Simulations

In our analysis we examine two potential implementations of the BF Radar device. The first (Low Fidelity) is realized in software and is being used in another study. The second (High fidelity) is realized in hardware, and has been partially built.

Low Fidelity (LoFi) Simulation Figure 2 shows the general layout (not the details) of MENDS, a low-fidelity simulation of this system. The system is implemented in Unity. The front panel (top image) shows the subsystems and the lights in the upper right corner of each square shows which subcomponents are working. An individual tray (bottom image) shows a tray and the components that are working (yellow light and white) and the components that are not working (red light and grayed out). This system has been briefly reported before (Ritter, Tehranchi, Brener, et al., 2019).

To troubleshoot the task (the details are in Table 1) the user clicks for the next problem, examines the lights, clicks on a tray, and examines its contents. They must then choose the broken component by clicking on it and clicking done.

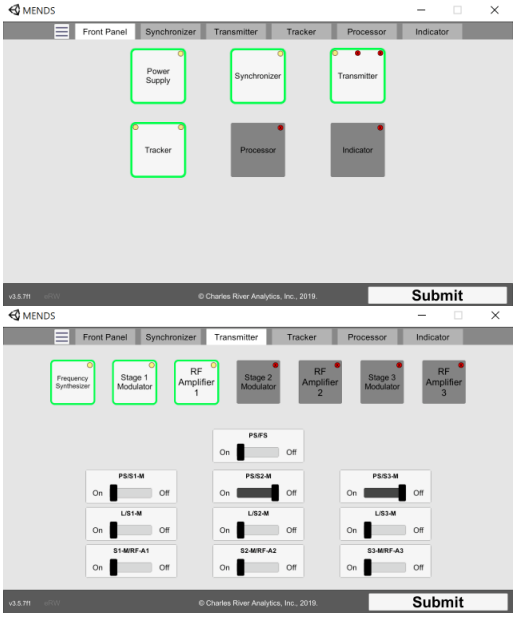


Figure 2. The initial interface of the MENDS low-fidelity task (top) and a tray (bottom). In these pictures, the fault is in the Processor subsystem.

High Fidelity (HiFi) Simulation Figure 3 depicts the higher fidelity version of this system. This system is realized in an approximate 2 ft physical metal and component cube using Raspberry Pi's and in Unity 3D. It has a cabinet holding trays for each subsystem. The top tray provides a summary of the system, including the indicator lights. The other trays each hold one subsystem.

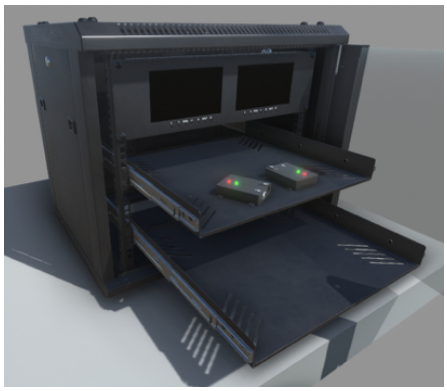


Figure 3. The high-fidelity realization of the BF Radar, showing the cabinet (with its door removed), two racks that will hold one subsystem each, and two components that will be inserted into component holders on the tray when the tray is built-out.

To troubleshoot the task (as an overview, the details are in Table 1) the user must first put on a grounding strap, and then examine the lights, open the cabinet door, pull out a tray, and examine its contents. They must then choose the broken component, find the replacement part, and replace the broken component. To set up a trial, the experimenter must have the user look away, replace a working component with a broken component, and then close the tray and the door.

The Task Assumptions

We assume that the user has been taught the BF Radar schematic and has it available, either in their head or on a sheet of paper. Table 1 shows the subtask times and the total time. The times are broken up into Learnable and Fixed tasks. The learnable tasks improve with practice; the fixed do not. These steps and their times (similar to and taken from the Keystroke-Level Model, Card, Moran, & Newell, 1983) are shown in Table 1. The user will start with the front display panel, and will have to examine the lights to know what tray and component to examine. Each step takes time, and we assume is error free. We use the Overdriven Amplifier, an early component in the system, as the example fault for this analysis.

The Learning Theory

The time to perform a task is broken down into two types of time: skills to be learned and skills that are already learned (or, essentially learned). Skills that are to be learned get improved with practice. On this task, learnable skills include: recognizing the lights and their implications. Skills that are essentially already learned are moving the mouse and clicking, and system response times include replacing faults or inserting faults by the system.

These times are used to compute the time to do the task using Eq. 1. This equation is consistent with Soar's (Newell, 1990) and ACT-R's (Anderson, 2007; Ritter, Tehranchi, & Oury, 2019) learning theories, and the learning curve in general (Ritter & Schooler, 2001). The times are computed in 10-minute blocks. Thus, the first 10 min. block of trials are done at 2.7 (LoFi) and 1.0 (HiFi) trials per minute, then in the second block the pace is updated to reflect what is learned after 10 min. This is repeated for nine more blocks.

$$\text{Time} = \text{Fixed tasks} + \text{learned tasks}\ (\text{Trial})^{-\alpha} \quad (1)$$

The choice of α (alpha) was arbitrarily chosen as 0.2. This value of α is consistent with values from Newell and Rosenbloom (1981, 0.06 - 0.81, a variety of tasks); and similar to values from Delaney, Reder, Staszewski, and Ritter (1998, 0.265 - 0.510, mental arithmetic); and Kim and Ritter (2016, 0.4 – 1.2, spreadsheet tasks).

We also looked at the time if users were to move back to the HiFi trainer at the end of each 10-min. block. That is, if a user were to train on the LoFi simulator and then move to the HiFi simulator at the end of each block. This curve is thus not a learning curve, but shows how well the learner would perform in the HiFi simulator after that much practice in the LoFi simulator. Equation 2 shows how that time is computed. We include the power law effect for the new task, but on the subtasks in the HiFi simulation, they have not been learned, and thus they are trial 1. This is just the subtask time itself, no learning has occurred.

(2) Time = Fixed tasks(Hi) +
Learnable tasks only in Hi (1) $^{-\alpha}$ +
(learnable tasks in both) (Trial) $^{-\alpha}$

Table 1. Task analysis. The fault modeled is the Overdriven Amplifier fault. (times are in s.)

		MENDS Low Fidelity		3D High Fidelity	
		Skills		Skills	
		Learnable	Fixed	Learnable	Fixed
	Sub total time (s)	15.75	7.55	32.75	26.00
	Total	23.30		58.75	
	Trials per min.	2.6		1.0	
Step	Setup				
0	Insert fault		0.05	12	0
		click "Next"		insert fault	
1	Approach System		1.5		5
2	Open cabinet	1	1.5	1	3
		click to open		lift lid	
3	Ground yourself	0	0	4	6
		not required			
	Find subsystem				
4	Check front panel	5.25		5.25	
	scan first panel, 0.25 per light plus Mop				
5	Choose tray	1.5	0	1.5	0
		Mop			
6	Open tray	1	1.5	2	4
		move+click		reach etc.	
	Tray loop				
7	Check light	0.25		0.25	
		scan first tray, 0.25 per light			
8	If off, have answer	1.5		1.5	
	If on, go to next light				
9	Check light	0.25		0.25	
	Overdriven amp fault				
10	Recognize fault	2		2	
	Replacement				
11	Indicate fault	1.5	1.5	1.5	6.5
		Mop+move+click		replace physically	
	Confirm done	1.5	1.5	1.5	1.5
		Mop+move+click		say "done"	

Simulation Results

Table 2 shows the number of repetitions of the tasks that arise across the ten 10-minute blocks. The LoFi simulator has a much larger number of reps, and this difference is maintained across the total training time, although the ratio between HiFi and LoFi decreases with practice.

Table 2. Number of total task repetitions over ten 10-min. blocks.

	Total Repetitions		
Block	LoFi	HiFi	Ratio
1	26	10	0.26
2	64	23	0.24
3	105	37	0.23
4	149	51	0.22
5	194	66	0.22
6	240	81	0.22
7	287	96	0.22
8	334	111	0.22
9	382	127	0.22
10	431	142	0.22

Figure 4 shows the learning curves for the HiFi and LoFi simulations, in linear and log-log coordinates. There is an additional line showing the response time for a user that practiced with the LoFi simulator and then moved to the HiFi simulator.

The plots show that the low fidelity users would get extremely fast on the material that is taught (green triangle, dashed line) compared to the high fidelity (blue square, solid line). The intercepts are different; the low-fidelity group starts out faster. And the slopes are different (best shown in the log-log plot), the low-fidelity group learns at a faster rate because they get an increasingly large number of repetitions because they are using a faster interface. With increasing practice, the low fidelity group remains faster, but the difference decreases as the power law effect is applied; that is, it takes increasingly larger amounts of practice for decreasing gains.

But, where would the new learners be on the whole task (HiFi) if they move to the HiFi after working with the LoFi simulator? When the low-fidelity group moves back to the high fidelity interface (black circle, dotted line) the effect of practice with the LoFi simulator is most pronounced early on. The black line shows not practicing all the tasks in the HiFi simulator can lead to faster times, but that this effect decreases with practice. And, if there was one or several learnable tasks in the HiFi task that were not in the LoFi task, the LoFi transition line could conceivably come in higher than the HiFi task at some point.

Human Participant Data that We Have So Far

We have three sets of data related to this task. On the original Laser Bank task, Ritter and Bibby (2008) saw reaction times ranging from 20 s initially to around 7 s when practiced. Friedrich and Ritter (2020) reported similar times.

In the MENDS task (LoFi interface), Ritter et al. (2019) saw a subject with 10 minutes of practice that went from 60 s to 22 s. The initial trials were thus slower, but the final time after 10 min. is approximately accurate.

We are currently running a study that will gather more data on the low fidelity version. We have run 8 out of 115 human participants so far.

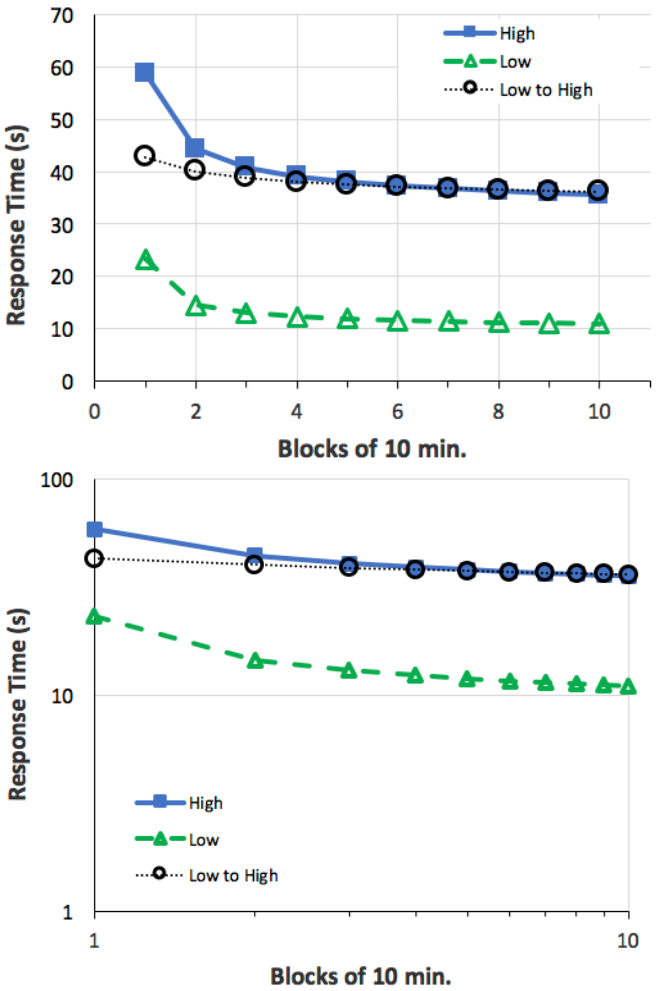


Figure 4. Response time for the 10 blocks for the high, low, and transitioned training schedules (linear, top; log-log, bottom).

Discussion and Conclusion

This analysis can help explain why there are still discussions about whether to choose low or high fidelity simulations. The analysis is sensitive to different assumptions about time costs of the two training systems. The analysis shows that the expected factors influence the amounts of learning: previous training on a task, setup costs for a training task, what can be transferred, what can be trained, and how much training is required. Each will influence the learning curves and the differences between the two levels of fidelity. This analysis points out that it is probably worthwhile to note and document what tasks are being trained in each system, and how many repetitions they are getting.

What is also clear is that time to train is an important measure. When there is a lot of training time (i.e., a large number of training trials is available), a low fidelity trainer does not offer as much benefit as when training time on the full system is limited. If a low fidelity trainer is available, it might not so much save time but save money (or lives or equipment if the situation being trained is dangerous). Lower fidelity training systems, if they cost less, can also lead to large learning gains even when transitioned to more complex tasks, and this has been seen before (Alluisi, 1991; Caro, Isley, & Jolley, 1973). It would be interesting to put those situations into these analyses.

This approach thus offers a calculus, a way for choosing how and why to use different levels of simulations. It can provide support for how much more training can be obtained from each type of simulation.

It could also be used to avoid the awkward situation where spending effort to make the simulation/training more faithful to the external environment by including behavior that is not greatly influenced by learning would none-the-less lead to learning less. Tasks that do not get faster and do not get learned are cases where fidelity could be dropped.

In this task, there does not appear to be a cost to starting low and going to the high fidelity training situation. This approach can save substantial time and resources. This approach shows that for this task there appears to be no cost to starting low and going high, unless there are essential skills that are learned and that are not in the LoFi simulator. Putting on a grounding strap, for example, if it was learnable and not taught in the LoFi simulator, could have an important role in this story.

We have run this analysis of the grounding strap as an example task only in the HiFi task. The curve indicates that in the first few training blocks, the LoFi interface still leads to faster performance. As amount of training increases, there is a cross-over point where the low fidelity performance is dominated by not having practice on the unpracticed task, and when the user transfers to the HiFi task, they are slower than the full task for the same training time. This effect should be explored further.

More Repetitions Are Important Early

The analysis shows that if you have only a short period to train, it is better to have learners on a low fidelity system. Figure 4 shows that the low fidelity when transitioned back to high was faster than only high fidelity because the learner had more practice on what could be learned. Performing more repetitions in the same period of time has a greater effect on learning when there is not a lot of trials. On the other hand, at larger amounts of practice, learners on the low fidelity do not gain as much relative to the high fidelity as they do at low practice time. The low fidelity is still faster, but the effect is smaller. In some situations this will still greatly matter (where differences in response time are important, such as adversarial tasks), and in some situations this will not (perhaps in safety tasks where doing the task correctly and slow is good enough).

Limitations

There are several limitations to this analysis. We have revised the task in Table 1 numerous times. Thus, there are likely to still be some inconsistencies. The model's general predictions appear to be robust against these changes, how-

ever. As we updated Table 1 while writing this paper, the curves in Figure 4 did not substantially change.

These analyses do not account for other differences in training systems such as cost, risk to the learner, environment, and equipment, time to get to the system, and so on. These are important considerations, and will have an important impact on training system choice.

Future Work

As a next step we are moving this analysis to R and doing a more detailed analysis. We will examine different tasks (faults) as well. We continue to run the study of the LoFi condition. The physical apparatus will provide more detailed empirical results to support this approach.

There are several analyses that we would like to do in the future. We would like to explore what happens when there are more tasks that are not trained in the LoFi simulation. This may lead to a situation where coming back to the HiFi task from the LoFi simulator is slower than staying on the HiFi curve. There currently is the use of the grounding strap as an example subtask. There could be numerous tasks like this in other situations. Other considerations such as cost may also be important.

We would like to generate a set of plots showing the effect of changes in learning rate (e.g., 0.1 to 1.0 in 0.1 steps). Exploratory analyses show that higher learning rates can alter the curves and the relative value to each level of simulator fidelity. It would also be interesting to see the net cost of the LoFi curves, either in training time or training costs.

It would be useful to make this analysis even easier to use. It could then be used to analyze more realistic, complex tasks, for example, as IMPRINT does (Booher & Minninger, 2003). This analysis could include the costs of building the additional LoFi interface. This tool could even go so far as to predict the cost of each component in the LoFi interface (e.g., building it out more could cost a little more but lead to greater learning savings, system saving, or system effectiveness). This approach can also be informed by tools to model users in interfaces automatically (John & Jastrzembski, 2010; Wallach, Fackert, & Albach, 2019), and could be potentially included in them.

This work can lead to a better method to determine optimal simulator training time based on examining performance improvement through using learning curves.

References

Alessi, S. M. (1988). Fidelity in the design of instructional simulations. *Journal of Computer-Based Instruction, 15*(2), 40–47.

Alessi, S. M., & Trollip, S. R. (1991). *Computer-based instruction: Methods and development*. Englewood Cliffs, NJ: Prentice-Hall.

Allen, J. A., Hays, R. T., & Buffardi, L. C. (1986). Maintenance training simulator fidelity and individual differences in transfer of training. *Human Factors, 28*(5), 497-509.

Alluisi, E. A. (1991). The development of technology for collective training: SIMNET, a case history. *Human Factors, 33*(3), 343-362.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York, NY: Oxford University Press.

Booher, H. R., & Minninger, J. (2003). Human systems integration in Army systems acquisition. In H. R. Booher (Ed.), *Handbook of human systems integration* (pp. 663-698). Hoboken, NJ: John Wiley.

Caro, P. W., Isley, R. N., & Jolley, P. B. (1973). *Research on synthetic training: Device evaluation and training program development* (Technical Report 73-20). Fort Rucker, AL: Human Resources Research Organization (HumRRO), Division No. 6 (Aviation).

Charvat, G. L. (2011). MIT IAP 2011 laptop based radar: Block diagram, schematics, bill of material, and fabrication instructions. from https://ocw.mit.edu/resources/res-ll-003-build-a-small-radar-system-capable-of-sensing-range-doppler-and-synthetic-aperture-radar-imaging-january-iap-2011/

Dahlstrom, N., Dekker, S., van Winsen, R., & Nyce, J. (2009). Fidelity and validity of simulator training. *Theoretical Issues in Ergonomics Science, 10*(4), 305–314.

Delaney, P. F., Reder, L. M., Staszewski, J. J., & Ritter, F. E. (1998). The strategy specific nature of improvement: The power law applies by strategy within task. *Psychological Science, 9*(1), 1-8.

Friedrich, M. B., & Ritter, F. E. (2020). Understanding strategy differences in a diagrammatic reasoning task. *Cognitive Systems Research, 59*, 133-150.

Havinghurst, L. C., Fields, L. E., & Fields, C. L. (2003). High versus low fidelity simulations: Does the type of format affect candidates' performance or perceptions? In *27th Annual IPMAAC Conference on Personnel Assessment: Exploring New Horizons in Assessment*.

John, B. E., & Jastrzembski, T. S. (2010). Exploration of costs and benefits of predictive human performance modeling for design. In *Proceedings of the 10th International Conference on Cognitive Modeling*, 115-120. Philadelphia, PA: Drexel University.

Kieras, D. E., & Bovair, S. (1984). The role of a mental model in learning how to operator a device. *Cognitive Science, 8*, 255-273.

Kim, J. W., & Ritter, F. E. (2016). Microgenetic analysis of learning a task: Its implications to cognitive modeling. In *Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016)*, 21-26. University Park, PA: ACS Lab, Penn State.

Lesgold, A. M., Lajoie, S., Bunzon, M., & Eggan, E. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin, R. Chabay & C. Scheftic (Eds.), *Computer assisted instruction and intelligent tutoring systems: Establishing communication and collaboration*. Hillsdale, NJ: Erlbaum.

Liu, D., Macchiarella, N. D., & Vincenzi, D. A. (2009). Simulation fidelity. In *Human factors in simulation and training* (pp. 61-73): Taylor & Francis.

Miller, R. B. (1954). *Psychological considerations in the design of training equipment*. (WADC Report No. 54-563, AD 71202). Springfield, OH: Carpenter Litho & Prtg. Co.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 1-51). Hillsdale, NJ: Erlbaum.

Noble, C. (2002). The relationship between fidelity and learning in aviation training and assessment. *Journal of Air Transportation, 7*, 34–54.

Prophet, W. W., & Boyd, H. A. (1970). *Device-task fidelity and transfer of training: Aircraft cockpit procedures training* (Technical Report 70-10). Fort Rucker, AL: Human Resources Research Organization (HumRRO), Division No. 6 (Aviation).

Ritter, F. E., & Bibby, P. A. (2008). Modeling how, when, and what is learned in a simple fault-finding task. *Cognitive Science, 32*, 862-892.

Ritter, F. E., & Schooler, L. J. (2001). The learning curve. In W. Kintch, N. Smelser & P. Baltes (Eds.), *International encyclopedia of the social and behavioral sciences* (Vol. 13, pp. 8602-8605). Amsterdam: Pergamon.

Ritter, F. E., Tehranchi, F., Brener, M., & Wang, S. (2019). Testing a complex training task. In *Proceedings of the 17th International Conference on Cognitive Modeling (ICCM 2019)*, 184-185.

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2019). ACT-R: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science, 10*(3), Paper e1488.

Swezey, R. W., Perez, R. S., & Allen, J. A. (1991). Effects of instructional strategy and motion presentation conditions on the acquisition and transfer of electromechanical troubleshooting skill. *Human Factors, 33*(3), 309-323.

Thorndike, E. L., & Woodworth, R. S. (1901). The influence of improvement in one mental function upon the efficiency of other functions Part I. *Psychological Review, 8*(247–261).

Wallach, D. P., Fackert, S., & Albach, V. (2019). Predictive prototyping for real-world applications: A model-based evaluation approach based on the ACT-R cognitive architecture. In *DIS '19: Proceedings of the 2019 on Designing Interactive Systems Conference*, 1495–1502.

Interactive Grounding and Inference in Instruction Following

Dario D. Salvucci (salvucci@drexel.edu)
Department of Computer Science, Drexel University
3141 Chestnut St., Philadelphia, PA 19104, USA

Abstract

Learning by instruction is one of the most common forms of learning, and a number of research efforts have modeled the cognitive process of instruction following, with many successes. However, most computational models remain brittle with respect to the given instructions and lack the ability to adapt dynamically to variants of the instructions. This paper aims to illustrate modeling constructs designed to make instruction following more robust, including (1) more flexible grounding of language to execution, (2) processing of instructions that allows for inference of implicit instruction knowledge, and (3) dynamic, interactive clarification of instructions during both the learning and execution stages. Examples in the context of a paired-associates task and a visual-search task are discussed.

Keywords: Instruction following; cognitive architectures; cognitive code; interactive task learning.

Introduction

Learning by instruction can be defined, in simplified terms, as the process by which a teacher provides a learner with instructions for a task and the learner follows the instructions to perform the task. The process of learning by instruction has been a focus of numerous cognitive-modeling efforts in past decades, such as those using ACT-R (e.g., Anderson et al., 2004; Salvucci, 2013; Taatgen & Lee, 2003; Taatgen, Huss, Dickison, & Anderson, 2008) and Soar (e.g., Howes & Young, 1997; Huffman & Laird, 1995; Lewis, Newell, & Polk, 1989). More recently, there has been an increased focus on interactive task learning through natural interaction with a human instructor (see Laird et al., 2017; Kirk & Laird, 2014).

This research on instruction following has primarily focused on skill acquisition and improvements with learning over time. In contrast, less attention has been paid to the translation of instructions to knowledge: while past efforts have generally included a basic version of the translation process, this aspect of the models is often brittle and dependent on a very particular specification of instructions. For example, the model described in Salvucci (2013) accepts simplified textual instructions and would break easily with only slightly different instructions. Some efforts have aimed to make such a process more robust—for instance, by utilizing a more flexible knowledge representation (e.g., Taatgen et al., 2008) or by relying on dynamic interaction with the teacher (e.g., Laird et al., 2017). Nevertheless, the robustness of instruction following and learning continues to be a challenge for modern cognitive models and architectures.

This paper explores several ways in which instruction following can be made more flexible and robust. Specifically, this work examines three areas for improving robustness: grounding of instruction language to knowledge representations, inference of implicit instruction knowledge, and dynamic interaction to clarify or augment instruction knowledge. The models here have been developed using the Think architecture (https://github.com/salvucci/think) which uses a cognitive-code approach (Salvucci, 2016) to embed the theories and mechanisms of cognitive architectures (primarily ACT-R: Anderson et al., 2004) into a modern programming language (in this case, Python). The following sections provide an overview of the modeling approach, focusing on the instruction interpretation and execution processes, and then discuss the three areas of improvement along with a description of the associated models.

Modeling with Cognitive Code

Before exploring the fuller model of instruction following, first we present a basic model of a simple task to illustrate the workings of Think's cognitive code and how it contrasts with traditional production-system cognitive architectures. The sample task examined here is the so-called paired-associates task (Anderson, 1981): the participant reads a word, tries to recall and type a digit associated with that word, and then reads the associated digit, eventually learning the word-digit pairings. Cognitive code allows for a clean separation of task and model code, each of them running on separate threads and interacting via elements of the environment (e.g., a desktop-computer display, keyboard, mouse, speakers, etc.).

Table 1 shows the simple (Python) code that implements the task itself. The code first clears the display and then presents the word stimulus on the display, then waits 5 seconds (as dictated by the experiment). If the participant keys in a response, the response is logged for correctness; otherwise, if no response is keyed, an incorrect response is logged. The trial ends when the code presents the associated digit and waits another 5 seconds. These steps are repeated for the various stimulus pairings and across trial blocks.

The cognitive code representing the cognitive model, in Table 2, is similarly straightforward. The first line directs the vision module to wait for a particular stimulus of type *word*, and when it is found, encodes the visual object in the `word` variable. Each line of code incurs a passage of virtual time that aligns temporally with the task code (as well as any other cognitive threads that may be running; see Salvucci & Taatgen, 2010). The next few lines attempt to recall a memory chunk that associates the word with its associated digit, and if successful, the model types the digit. Finally, the model encodes the visual digit and stores the association between word and digit in memory.

Because cognitive code is grounded in a modern programming language familiar to most programmers, learning to write models under this approach is much easier than with

Table 1: Paired-associates task code.

```
self.display.clear()
self.display.add_text(50, 50, word, isa='word')
self.wait(5.0)
if not self.responded:
    self.log('incorrect response')
self.display.add_text(50, 50, digit, isa='digit')
self.wait(5.0)
```

Table 2: Paired-associates model code.

```
visual = self.vision.wait_for(isa='word')
word = self.vision.encode(visual)
chunk = self.memory.recall(word=word)
if chunk:
    self.motor.type(chunk.get('digit'))
visual = self.vision.wait_for(isa='digit')
digit = self.vision.encode(visual)
self.memory.store(word=word, digit=digit)
```

production systems, which are not nearly as familiar to programmers today. Cognitive code also takes advantage of common programming idioms—for instance, returning `None` for a failed memory retrieval, compared to a more complex production-system method of handling such a failure. On the other hand, production systems offer some of their own benefits, such as a more flexible partial ordering of execution. Nevertheless, cognitive code aims to provide the major advantages of cognitive modeling—broadly speaking, accounting for and predicting cognitive, perceptual, and motor performance—to a wider programming audience with as low barriers as possible to getting started in the modeling process.

Instruction Interpretation and Execution

We now take the next step in our modeling, moving from single-task models (like the one above) to a more general model that takes instructions and can execute a variety of tasks. The model of instruction following proposed here can be characterized at the highest level in terms of two stages: interpretation and execution. The interpretation stage involves translating the given instructions into a mental representation that encodes the necessary cognitive, perceptual, and motor actions that combine to perform the desired task. The execution stage involves recalling each instruction and then actually performing the cognitive, perceptual, and/or motor actions involved. Our model also aims to account for a realistic passage of time through both stages—most notably, processing instructions step by step over time (as opposed to assuming an already-encoded full set of instructions in memory).

The interpretation stage is implemented in the model as follows. Each instruction is assumed to be spoken aloud by the teacher such that the model can, like an experiment participant, hear and process the information incrementally. (Alternatively, we could assume that each instruction is presented on-screen to the participant; the model would behave largely the same except for utilizing visual instead of aural channels.) The model then interprets each instruction by attempting to understand its meaning and converting it to an associated mental representation. Because of the real-time nature of how the model receives instructions, the decay in the architecture's memory system—again, based on ACT-R—necessitates some practice of these instruction chunks so that they can be properly recalled in the next stage.

For example, consider the instructions in Table 3 for the paired-associates task. When interpreting these instructions, the model starts with the first statement—'*To perform a task*'—and understands that what follows are instructions for this particular task. Then, the model translates each step of the instructions to one or more relations, implemented as ACT-R-like chunks—for example, `WaitFor(word)` or `If(Recall(digit, word), Type(digit))` for the first two steps in Table 3. Each of the chunks is boosted in memory to ensure later recall.

Table 3: Sample instructions for the paired-associates task.

To perform the task
Wait for a word
If you can recall the digit for the word, type the digit
Wait for a digit
Remember the word and the digit
Repeat

The execution stage then uses the stored mental representations to perform the given task. At each step, the model recalls the chunk(s) for that step and performs the actions associated with the step—for instance, the `WaitFor(word)` chunk would invoke the visual system in waiting for a visual stimulus, and when found, the model would note the encoded object as the *word*. In doing so, the model builds up a context such that it may use information later (such as when the word and digit need to be remembered together in the sample task).

While our description above might suggest that interpretation and execution are two discrete stages that occur one after another, in fact these two stages are often interwoven: partial instructions might be provided so that a learner can practice a subgoal of the task; the learner may forget certain instructions and need to refresh their memory; the learner may also realize that their mental representation is ambiguous or deficient in some way and need clarification during execution; and so on. Such examples will be expanded further in the next section.

Interactive Grounding and Inference

The above description of instruction following as interpretation and execution are quite general; however, a simple

straightforward implementation of these processes may yield a model that is very fragile with respect to the instructions. The predecessor to this work (Salvucci, 2013) focused on a model that could account for behavior across a wide range of tasks, and did not emphasize aspects of instruction flexibility; the model had only a minimal interpreter for the simplest pseudo-English instructions (e.g., '*Wait-for visual-change*'). At the same time, the general problem of natural-language understanding with respect to instruction following is of course an extremely difficult problem in its own right, and such a general model is not currently feasible. Thus, our primary aim is to develop a model that minimizes the most general natural-language challenge but still allows for as much flexibility and robustness as possible. We now describe severals ways in which we can generalize the previous approach, including incorporation of ideas from other efforts into a single integrated account of instruction following.

Instruction Grounding

One critical aspect of the interpretation stage can be characterized as instruction grounding in which the natural-language statements and their subcomponents are grounded to objects and actions in the real world. Several recent efforts in the Soar community in particular have made significant strides in this area (e.g., Lindes, Mininger, Kirk, & Laird, 2017; Mohan, Mininger, Kirk, & Laird, 2012). For example, the Lucia system incorporated into the Rosie agent (Lindes et al., 2017) provides grounding for simple objects (e.g., '*the green rectangle*'), prepositional phrases (e.g., '*the green square to the left of the blue square*'), and whole sentences; in doing so, Rosie can learn tasks (in this case, simple games) and uses the grounded knowledge to reason about and act upon the associated objects in the world.

We follow a similar approach here, interactively receiving the instructions in sequence and incrementally grounding each component. Consider the paired-associate instructions presented earlier in Table 3. The parser implemented in Think takes a natural-language phrase such as '*Wait for a word*' and builds a declarative memory chunk `WaitFor(word)` as a mental representation of the phrase. The concept of *word* is grounded to the next visual object that appears to the model, and the model will store the mapping from *word* to this object in the current context (equivalent to ACT-R's imaginal buffer; see Anderson et al., 2004). In other cases where a specific visual object is referenced (e.g., on a crowded screen), the model allows the (virtual) experimenter to "point out" visual information—for instance, hearing the phrase '*Read the letter*' while pointing at the object—which gives the model an associated visual point along with the verbal information (Salvucci, 2013). Later in the model's simulation run, actions such as *Wait for* or *Read* will be grounded to their respective psycho-motor actions during the execution stage.

Sometimes, ambiguity in grounding can arise when the same object is referred to by different words or phrases. For example, consider a case in which a teacher directs the learner to '*Wait for a digit*' and then later to '*Type the number*'. From the context, and in this case the lack of any other realistic interpretation, a human participant could understand the change and ground both *digit* and *number* to the same object, whereas a simpler model interpreter could not make this leap. The model here allows for the inclusion of potential synonyms in its declarative memory, which are assumed to be part of a person's general knowledge (not something learned during the task). When a term is discovered that cannot be grounded—e.g., '*Type the number*' when the model has not yet seen a *number*—the model checks for other potential interpretations within its existing context. In our example, the model will have already grounded the *digit*, and thus it can search for and find an interpretation whereby *number* and *digit* are the same object.

Table 4: Sample instructions for the visual-search task.

(a)	To perform the task Find the 'C' Move the mouse to it Click on it Repeat
(b)	To perform the task Find the 'C' Click on it Repeat
(c)	To perform the task Click on the 'C'

Another common and useful aspect of natural language for instructions arises in anaphora resolution, or more specifically, pronoun resolution. Table 4(a) provides sample instructions for a visual-search task in which the participant finds the letter 'C' among a set of distractors (such as the letter 'O'). Pronoun resolution—specifically here, resolving the meaning of the word *it*—allows for two major benefits: first, it allows for more natural expressions of the instructions of the part of the teacher; and second, it allows the model to ground multiple references to the same physical object (in this case, the same 'C' mentioned earlier in the instructions). Much like the model of Lindes et al. (2017), the model here builds up a representation context incrementally, first noting that there is an object 'C', and later noting that *it* must refer to this earlier object. Admittedly, pronoun and anaphora resolution are much more complex in the general case; however, even the straightforward method here covers many simpler cases and already nicely enhances the flexibility of the model's parsing and interpretation.

Instruction Inference

Beyond the language of the instruction steps, some of the variability from a teacher's instructions arises in inclusion or exclusion of the steps themselves. In particular, some steps

may be explicitly stated in one circumstance but only implicitly suggested in another; in the latter case, the model must infer any intermediate instructions or actions. Table 4 includes three alternative sets of instructions: (a) long-form instructions that explicitly direct the participant to *Find*, *Move* to, and finally *Click* on the desired target, plus an explicit *Repeat* step; (b) shorter instructions that skip the *Move* step; and (c) even shorter instructions that only direct the participant to '*Click on the 'C'*' without any other steps. In each case (and one might easily imagine further alternatives), we would expect the same behavior from the learner.

Our approach to this challenge represents a blend of two key ideas in earlier work. First, more recent models of instruction following developed in ACT-R (e.g., Taatgen et al., 2008) have encoded instructions along with a set of preconditions and postconditions, such that an instruction step may execute only when its preconditions have been satisfied, and its execution then results in postconditions that may in turn be needed by other steps. Second, the Soar cognitive architecture (Laird, Newell, & Rosenbloom, 1987) has as one of its core principles the idea of resolving an impasse: when the next action cannot be easily determined, the architecture generates an impasse and creates a subgoal to resolve this impasse. The model borrows the spirit of each approach in the execution of instructions. Certain actions, such as clicking the mouse on an object, have a natural precondition, such as moving the mouse to that object. When attempting to follow such an action, if the precondition is not met, the model first tries to execute a subgoal that will resolve that precondition, which could potentially trigger another subgoal. In Table 4(c), the *Click* action requires the *Move* action, which in turn requires the *Find* action—and thus the single instruction '*Click on the 'C'*' triggers the same sequence of actions as the equivalent three steps in Table 4(a).

Along these lines, similar simple inferences could be made in other ways for these instruction sets. For example, Table 4(c) omits the final *Repeat* step, but it would be reasonable to assume that if a participant would continue to be presented with similar stimuli, they would infer this repeat on their own, and the model does the same.

The approach here is not as general as Soar's impasse mechanism, since it does not claim to be a general approach to subgoaling; the approach is more akin to the precondition/postcondition work of Taatgen et al. (2008), although here, conditions are not stored in the declarative memory chunks but instead embedded in the execution processing of the individual actions. Our approach could also be viewed as a basic form of backward chaining seen in other systems (e.g., Langley & Choi, 2006). On a larger scale, the interaction between teacher and learner (described shortly) may lead to even more complex scenarios—for example, a teacher could modulate instructions based on learner's expertise, common and shared knowledge, and so on, leaving the learner to infer simple steps or perhaps to derive more complex inferences between new pieces of knowledge.

Interactive Learning and Execution

As stated earlier, although we have mostly emphasized separate interpretation and execution stages to this point, the reality of instruction following is often much more complex, involving interaction between teacher and learner throughout the learning process. This idea has been a focus of the work on interactive task learning (see Laird et al., 2017), in which "the learner actively tries to assimilate the meaning of the instruction while performing the task, and learning occurs in conjunction with that task's performance." During the interpretation stage, a learner might stop when confused by an instruction and ask the teacher what is meant by that instruction. During the execution stage, the learner might realize that there is actually ambiguity where they did not anticipate (e.g., two words on the screen when looking for a *word*).

The current model is built with interactivity in mind, allowing for a stream of communication between the (simulated) teacher and the (model) learner. The teacher provides verbal instructions to the model, and the model performs the task over time—but at any stage, either of them may interact with the other to communicate questions or information (see the description of "communicative grounding" in Chai et al., 2018). For example, consider a situation for the paired-associates task in which the model remembers a *digit* but then is instructed to '*Type the number*'. As mentioned earlier, the model has one avenue to solve this ambiguity, namely in recalling *number* as a possible synonym of *digit*. But what if this synonym pair was not known to the learner, or is a distant semantic relation that could not be easily inferred (e.g., *digit* and *target*)? If no synonym is available, the model stops and asks the teacher a question such as '*Which is the number?*', and waits for a response to process using its audition module. When the response is given—e.g., '*the digit*'—the model remembers this association and uses it for future processing. (The association might even be forgotten if the memory chunk decays too much before its next use, which would trigger the model to repeat the question to the teacher.) In this way, the model gains additional ways to augment its understanding and clarify ambiguities and/or gaps in its knowledge.

Discussion

This paper has provided an overview of several ways in which computational models of instruction following can be made more flexible and robust with respect to variations in the instruction and learning process. Our discussion of the relationship to human data has been somewhat non-traditional for a cognitive-modeling effort: we have generally argued that human participants can adjust to these variations and then shown that the model can do the same. From a more traditional perspective, we can note that these models do indeed provide a reasonable fit to human performance. For example, Figure 1 shows the results of 10 simulations of the paired-associates model along with the results from human participants over blocks of trials (Anderson, 1981). The model fits the human data well for both correctness, $R = .98$, $RMSE = .08$, and

response time, $R = .99$, $RMSE = .17$, with ACT-R memory theory driving the predictions. In fact, for any of the variants of the instructions described here, the model results would be largely the same; small differences might arise due to extra cognitive processing of, for example, synonyms or interactive communication, but the qualitative behavior and fit of the model would not change in a significant way.

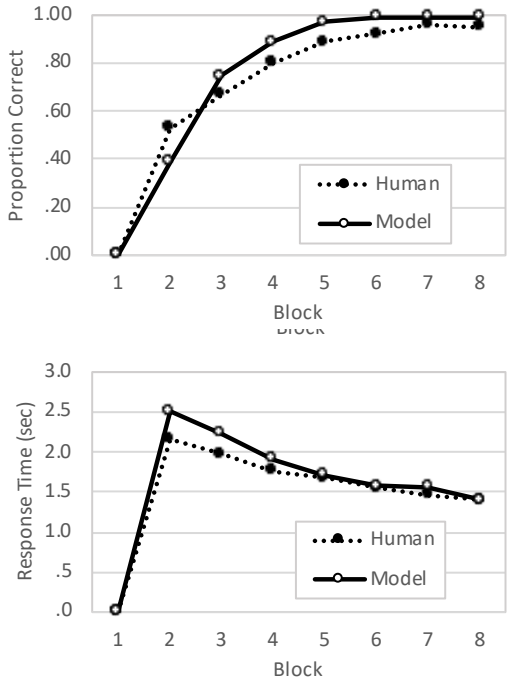


Figure 1: Human and model results for the paired-associates task showing (a) correctness and (b) response time.

Thus, unfortunately our current human data does not allow us to fully validate the techniques here. To address this challenge, our current work is part of a larger effort to develop an *undifferentiated agent* which is not specific to any one task but instead can take instruction and then perform a wide range of tasks (see, e.g., Salvucci, 2013). We are working toward applying such an agent to a battery of tasks, addressing various challenges along the way, especially with respect to the types of inference (or "gap-filling") that might be done more robustly with a fuller knowledge ontology and reasoning system. This effort aims to provide dual benefits of deeper understanding of human behavior and broader development of systems for practical applications such as synthetic teammates (e.g., Myers et al., 2018).

Acknowledgments

This work was funded in part by a grant from the Air Force Office of Scientific Research (#FA9550-18-1-0371).

References

Anderson, J. R. (1981). Interference: The relationship between response latency and response accuracy. *Journal of Experimental Psychology: Human Learning and Memory*, *7*(5), 326.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036.

Chai, J. Y., Gao, Q., She, L., Yang, S., Saba-Sadiya, S., & Xu, G. (2018). Language to action: Towards interactive task learning with physical agents. In *IJCAI* (pp. 2–9).

Howes, A., & Young, R. M. (1997). The role of cognitive architecture in modeling the user: Soar's learning mechanism. *Human–Computer Interaction*, *12*(4), 311–343.

Huffman, S. B., & Laird, J. E. (1995). Flexibly instructable agents. *JAIR*, *3*, 271–324.

Kirk, J. R., & Laird, J. E. (2014). Interactive task learning for simple games. *Advances in Cognitive Systems*, *3*, 13–30.

Laird, J. E., Gluck, K., Anderson, J., Forbus, K. D., Jenkins, O. C., Lebiere, C., ... others (2017). Interactive task learning. *IEEE Intelligent Systems*, *32*(4), 6–21.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*(1), 1–64.

Langley, P., & Choi, D. (2006). A unified cognitive architecture for physical agents. In *Proceedings of the National Conference on Artificial Intelligence* (Vol. 21, p. 1469).

Lewis, R. L., Newell, A., & Polk, T. A. (1989). Toward a soar theory of taking instructions for immediate reasoning tasks. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 514–521).

Lindes, P., Mininger, A., Kirk, J. R., & Laird, J. E. (2017). Grounding language for interactive task learning. In *Proceedings of the First Workshop on Language Grounding for Robotics* (pp. 1–9).

Mohan, S., Mininger, A. H., Kirk, J. R., & Laird, J. E. (2012). Acquiring grounded representations of words with situated interactive instruction. *Advances in Cognitive Systems*, *2*, 113–130.

Myers, C., Ball, J., Cooke, N., Freiman, M., Caisse, M., Rodgers, S., ... McNeese, N. (2018). Autonomous intelligent agents for team training. *IEEE Intelligent Systems*, *34*(2), 3–14.

Salvucci, D. D. (2013). Integration and reuse in cognitive skill acquisition. *Cognitive Science*, *37*(5), 829–860.

Salvucci, D. D. (2016). Cognitive code: An embedded approach to cognitive modeling. In *Proceedings of the 14th ICCM* (pp. 15–20). University Park, PA: The Pennsylvania State University.

Salvucci, D. D., & Taatgen, N. A. (2010). *The multitasking mind*. Oxford University Press.

Taatgen, N. A., Huss, D., Dickison, D., & Anderson, J. R. (2008). The acquisition of robust and flexible cognitive skills. *Journal of Experimental Psychology: General*, *137*(3), 548.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors*, *45*(1), 61–76.

Drive the Bus: Extending JSegMan to Drive a Virtual Long-Range Bus

David M. Schwartz (dms7225@psu.edu)[1]

Farnaz Tehranchi (farnaz.tehranchi@psu.edu)[2]

Frank E. Ritter (frank.ritter@psu.edu)[1]

College of Information Sciences and Technology[1]

Department of Computer Science and Engineering[2]

Penn State, University Park, PA 16802 USA

Abstract

ACT-R has been used to study human-computer interaction. By default, ACT-R models can only interact with interfaces written in Common Lisp. JSegMan has allowed ACT-R models to interact with external interfaces without modification. Currently, JSegMan has been used in conjunction with ACT-R's standard motor module, which cannot model common behaviors such as holding down keys, chording (pressing multiple keys at the same time), and multihand actions (e.g., moving the mouse with the right hand while pressing a button with the left). Extensions to ACT-R's motor module have been developed to address these issues and are included with ACT-R. Like the original motor module, the extensions can only interact with interfaces written in Common Lisp. This paper describes modifications to update JSegMan to work with ACT-R's motor extensions and demonstrates its usage by creating a model to play *Desert Bus*. Furthermore, the implication of running a model over many hours is explored.

Keywords: Cognitive architectures; ACT-R; Motor control; Chording.

Introduction

The embodied cognition-task-artifact triad states that behavior in an interactive environment is mediated by three factors: embodied cognition, the task a user is performing, and the artifact they interact with. Byrne (2001) proposes that using ACT-R (Anderson, 2007; Ritter, Tehranchi, & Oury, 2018) can assist human-computer interaction studies because ACT-R deals with the entire triad at once—the architecture handles the limits of cognition, the model encodes task knowledge, and an artifact is necessary to provide stimuli to the model and handle its output (key presses and mouse movements). However, ACT-R in its current form can only interact with special or heavily modified interfaces, making it difficult to study human-computer interaction.

JSegMan (Tehranchi & Ritter, 2018a, 2018b) offers a method of interacting with an interface external to ACT-R without modification. It detects visual features from a screenshot of the computer's display to provide ACT-R with stimuli. In addition, it allows a model's motor movements to control a computer's peripherals. However, this new level of interaction is limited to the default functionality of ACT-R's motor module and thus is limited in the behavior it can model.

By default, ACT-R is only capable of supporting serial motor action. Multiple motor commands can be queued together to simulate quick typing, but the architecture must process each keypress separately. This prevents the architecture from being able to press multiple keys at once, thereby making it impossible to type certain symbols (e.g., open and close parentheses because they require the shift key), use keyboard shortcuts, and play many video games. These issues were raised and addressed by during the development of a model to play space fortress (Bothell 2010). However, JSegMan has yet to incorporate the extended functionality. To determine how JSegMan must change, we created a model to play a simple game, *Desert Bus*.

Our experience in developing the model has led to several proposals on how to grow JSegMan. First, JSegMan should add commands (e.g., press and release) that mimic those available in the extended ACT-R motor module. Second, JSegMan can reduce its overhead (and improve model accuracy in dynamic task environments) by using ACT-R's remote procedural call interface. This work also raises questions about long-term behavior in cognitive architectures.

Background

This section discusses ACT-R's structure and various methods researchers have used to have it interact with external interfaces. Also, the game used as a task is described.

ACT-R

The ACT-R cognitive architecture (Anderson, 2007; Ritter, Tehranchi, & Oury, 2018) implements the fixed features of cognition as modules. The primary function of the architecture is controlled by the declarative and procedural modules. The declarative module manages factual memory (e.g., George Washington was the first president of the United States) encoded as chunks while the procedural module handles memory about performing actions (e.g., to turn on a computer, you have to press the start button), encoded as productions. The facts in declarative memory, actions in procedural memory, and stimuli the model sees determines how it behaves. What the model sees and how it acts within its environment are controlled by the perceptual and motor systems (spread across four modules), respectively. However, ACT-R has issues interacting with external interfaces and simulations (Schwartz & Dancy 2019; Schwartz & Ritter 2019).

ACT-R/PM

ACT-R's current perceptual and motor systems are based on ACT-R/PM (Byrne, 2001). The system assumes the model is viewing and interacting with a computer. ACT-R/PM adds four modules to the architecture: vision, motor, speech, and audition. ACT-R/PM's perceptual and motor modules have been merged into ACT-R and come as part of the standard release. This section will only discuss the vision and motor modules as the others are not pertinent to this project.

The vision module handles what an ACT-R model can see. It represents the screen as a collection of features that represent text, images, lines, buttons, etc. Features are mapped to chunks that represent where and what an object is. The visual-location buffer controls the *where* system and allows a model to query for an object's location. Once a feature is found, the model can shift its attention to it and encode the object via the *what* system controlled by the visual buffer. This creates a detailed chunk for the model to use.

The motor system provides support for using a virtual keyboard and mouse. It represents a user with two hands and allows procedural memories in a model to move the hands, mouse, and punch/peck mouse buttons and keys. The duration of hand and finger movements are estimated via Fitts' Law.

It is important to note that ACT-R/PM only works with special interfaces. ACT-R/PM was originally written in Macintosh Common Lisp (MCL) and only extracts features from interfaces created in a particular set of tools included with ACT-R/PM. ACT-R/PM has been partially generalized, allowing it to pull features from the ACT-R Graphical User Interface, across various Lisp implementations. However, the root of the problem remains—the interface still needs to be written in a compatible Lisp variant using the predefined structures.

Shortcomings of and Extensions to the Motor System. Two issues are present in ACT-R's motor system. First, ACT-R's motor system cannot perform concurrent inputs that are common in everyday computer usage. This issue is caused by state management within the motor module. The module has three states: preparation, processor, and execution. New motor commands can be queued when the preparation state is free. However, only one action can be executed at a time as the execution state handles commands serially. Furthermore, these states control actions for both hands; therefore, performing an action with one hand prevents the model from using the other. This implies that ACT-R cannot model video games that require the user to use both hands concurrently.

Second, the motor module does not support holding down keys. The motor module supports punches and pecks, each of which presses and then releases a given key. Together, these issues limit the types of interaction ACT-R can model.

These limitations prevent ACT-R models from pressing multiple keys at once—meaning the regular behavior users exhibit when typing capital letters and using keyboard shortcuts cannot be modeled. A common workaround is to assume that the model has an extended keyboard with buttons that represent chords. Thus, to give an ACT-R model the ability to use copy and paste shortcuts, dedicated buttons would be added to ACT-R's virtual keyboard to input Control-c and Control-v chords, respectively.

These weaknesses were exposed and remedied during the development of a model to play Space Fortress (Bothell 2010). Separate execution states were added per hand, allowing ACT-R to use both hands in parallel. Several motor commands were added to facilitate holding down and releasing keys such as hold-peck, hold-punch, hold-key, and release. The extended system signals both presses and releases, so new handlers were added to devices to enable them to detect key and mouse button releases. Finally, a new module, called motor-extension, was added that has two buffers that can query the activity of each hand.

Network Interfaces

Another method of getting ACT-R to interact with an external interface is via a network interface. The JSON Network Interface (JNI) (Hope, Schoelles, & Gray, 2014) allows visual objects and motor movements to be shared over a network connection. The interface generates chunks for the visual objects on screen, packs them into a JSON record, and sends it to an ACT-R model. A special module unpacks the packet and adds the information to the visicon (the list of visual features currently on screen), allowing ACT-R to work with the visual information as normal. Similarly, motor commands in ACT-R generate a packet that is sent to the interface, which can be used to update the interface's state.

New versions of ACT-R (7.6+) have incorporated similar functionality. They are based on a remote procedure call (RPC) system that allows multiple clients to request actions from a server running ACT-R. Therefore, an interface can connect to the server and send visual chunks for models to use. Additionally, the interface can watch for motor commands and act based on them.

Both JNI and ACT-R's RPC system assume an interface can be modified. The task interface must have several features added to it. First, it must manage the connection to either JNI or ACT-R's RPC server. Second, it must be able to convert visual information into visual location and encoded object chunks. Third, it must be able to simulate inputs based on those received from JNI or ACT-R. These modifications can be nontrivial and take time away from the core reason for using ACT-R, to study human cognition in a task.

Segmentation and Manipulation

Another method of providing interaction to external interfaces is by parsing the screen and manipulating inputs. Therefore, this approach aims to alleviate the issues present in ACT-R/PM and network interfaces by allowing the model to "see" what is on the screen and actually interact with it. SegMan adopted this approach (St. Amant, Riedl, Ritter, & Reifers, 2005). SegMan created visual features by taking a screenshot of the display and separating the pixels into groups based on color and location. Patterns were used to combine

groups that met modeler specified criteria. Finally, patterns and groups could be parsed to identify visual features such as images, buttons, and text. In addition, SegMan could simulate mouse movement, clicks, and key presses by interacting with the operating system.

SegMan was written in C and worked with Microsoft Windows 95/98/2000/XP. In addition, it was designed to be a general programmable interface, and thus worked with several architectures including ACT-R, Soar, and EPIC. Unfortunately, the system was not maintained and over time became less usable.

JSegMan (Tehranchi & Ritter, 2018a, 2018b) is a modern implementation based on the segmentation and manipulation approach. JSegMan works separately from ACT-R, feeding visual information to it and capturing desired motor commands from it. The vision system works by taking a screenshot of the computer's display and detecting features requested by a model. Models are augmented to have memories of what an object (e.g., a button) looks like. These memories store images to search for in an interface. Finding a feature is handled by template matching—a computer vision algorithm that separates the screen into patches and compares each patch to a template (or desired image) pixel by pixel. The patch with the highest similarity to the requested memory image is returned.

Motor control is handled by interacting with the operating system. A signal representing a model's interaction (e.g., a punch or peck) is sent to JSegMan, which relays the corresponding action to the operating system.

JSegMan has shown that older models must be modified to work with real interfaces. A model designed to perform the Dismal spreadsheet task (Kim & Ritter, 2015) was modified to use JSegMan (Ritter, Tehranchi, Dancy, & Kase, in press; Tehranchi & Ritter, 2018a). The Dismal task asks subjects to compute values in a spreadsheet given a fixed set of instructions; Emacs was used to display and modify the spreadsheet. Forcing the model to really interact with the interface revealed deficiencies in the model's logic. After fixing them, the modified models performed better than the originals.

Desert Bus

The video game *Desert Bus* was used as a task during this study. *Desert Bus* was created by Dinosaur Games and published by *Gearbox Software*; it is available for free and runs on Windows machines. It was developed for a charity event. The game is based off an unreleased game of the same name designed by Penn and Teller in 1998.

Desert Bus has the player drive a bus on a straight road through the desert connecting Tucson, AZ and Las Vegas, NV. The trip takes approximately eight hours to complete one-way, at which point the player earns one point and is instructed to turn around and drive back. This process continues endlessly. All the while, the bus drifts slightly to the right. If the bus drives off the road, it is towed back to the beginning (in real-time), the trip odometer, and points are reset. The game cannot be paused. The player controls the bus with the WASD keys; W is used to accelerate, A and D turn left and right respectively, and S applies the brakes. The player can also look around with the mouse and click to open the door to the bus and turn on/off the radio. Figure 1 shows the player's view from inside the bus.

Figure 1. The player's view from inside the bus.

Model

Figure 2 shows a flowchart of the model's decision cycle. The model begins by holding down the W key to accelerate. After that, it enters a looping decision cycle where it looks for the yellow dividing line in the center of the road (Land & Horwood, 1995; Land & Lee, 1994) and uses its position to determine if the bus should be realigned. A realignment will occur if the line has drifted past 857 pixels; this is the initial position of the dividing line at the start of the game. The A key is pressed to turn the steering wheel and realign the bus. If no adjustment needs to be made, the model fires a production that symbolizes the decision to drive forward (without adjusting steering) and then restarts the decision cycle. As the game occurs in real-time, the ACT-R model also runs in real-time.

The model takes advantage of the fact that the bus will only drift to the right (causing the dividing line to move to the left). Thus, the model only has to worry about moving left or forward. A more robust model would also consider moving to the right to make up for overcompensating for the drift and ending up on the wrong side of the road. Our model does not worry about this because no other vehicles appear in the game.

JSegMan handles finding visual targets and simulating keyboard inputs for the model. Visual searches are requested at the start of the decision cycle, so the model will always know where the dividing line has drifted since the prior decision. Following the example of the Dismal model, an ACT-R device was used to detect key presses and signal JSegMan on the behavior to emulate. JSegMan does not have a persistent connection to ACT-R. Instead, a JSegMan process must be started (and run to completion) for each action. Data is passed to JSegMan via command-line arguments. Data is received from it by parsing its output stream. Furthermore, when JSegMan is running, the ACT-R model is paused.

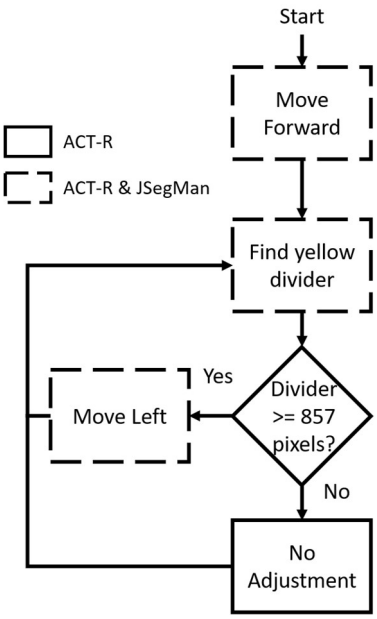


Figure 2. Flowchart of the model. Boxes with a solid border do not make use of JSegMan whereas boxes with a dashed border do. The model starts by driving forward. Then, it looks for the dividing line in the road and realigns the bus (by moving left) if the line has drifted far away.

The model only looks for the center dividing line, so it only has one template for JSegMan to look for, depicted in Figure 3. Templates in JSegMan are images, thus a screenshot of the game was used to generate the template.

Figure 3. Visual template used for the dividing line. The template was extracted from a screenshot of the game.

Finally, the model only handles driving. The player begins the game outside of the bus and must turn around and punch a timecard before entering the vehicle. To keep the model simple, we have a player punch the timecard, enter the bus, and then we start the model. Including these steps are obvious future tasks. Nevertheless, while undertaking the drive from Arizona to Nevada, it will be one of the longest running ACT-R models.

Demonstration Observations

Unfortunately, in its current state, the model is only able to drive for about a mile before being towed back to the beginning. The model always successfully makes one adjustment. However, the adjustment made is too large; it takes the bus from the extreme right edge of the road to the extreme left of the opposite lane. After that, the model will continue driving forward until the bus drifts back into the center of the road (between the two lanes). Then the model attempts to make another adjustment and over adjusts, driving off the road to the left.

The model fails to drive for more than a mile for a multitude of reasons. First, the template for the dividing line gets mismatched. The model only uses one template to identify the location of the divider. However, this template is not always satisfactory. As the bus drifts left and right across the road, the angle of the dividing line changes. When the bus is to the right of the divider, the angle is similar to that of the template and matches are more likely to be correct. However, when the bus over adjusts and ends up on the left of the divider, the template does not match as well. Furthermore, ACT-R is unaware of the quality of a match. JSegMan is used to find objects and features on the display. However, JSegMan does not return any information about the quality of a match, but a matching request will always return a position. Thus, a feature will always be found even if it is not present, meaning ACT-R does not know when it should avoid putting the feature in the visicon.

In theory, using multiple patterns could remedy the issue. Patterns of the divider at different angles would be a proxy for where the bus is, allowing the model to determine if an adjustment is necessary. However, this process would take too long. Currently, it takes 6.01 seconds on average (n=100) to match the divider template. Furthermore, this is about the time it takes for the bus to drift from the center of the road to the rightmost edge; therefore, if the model attempted to match a second template, it would drive off the road before having the chance to make an adjustment.

Additionally, the over adjustment is an artifact created by the overhead of running external processes. JSegMan does not have support for holding keys or presses of arbitrary lengths. To make up for this, a Java program was constructed to simulate key press and release events (to mimic the signals sent by ACT-R) and is invoked just like JSegMan. This program was used to determine what the effects would be of incorporating press and release commands into JSegMan. To simulate a full key press and release this program would have to be run twice, the former sending the press signal while the latter sent the release. According to the model, an adjustment involves a rapid peck lasting for 0.08 seconds. However, on average (n=100) this mechanism takes 2.79 seconds to simulate an input. Furthermore, the input seen by the operating system is longer than 0.08 seconds because of the time spent creating the release process. Using the newest version of ACT-R would help alleviate some issues (notably those for key presses/releases) by reducing overhead. Newer versions of ACT-R are remote procedure call based. If

JSegMan is modified to be a client to ACT-R's event dispatcher, it will not need to be restarted, reducing overhead to the time it takes to send several packets (representing the command to execute). This change will require JSegMan to rely less on the device, as newer versions of ACT-R try to avoid using it. However, this should not be an issue as JSegMan will also be able to query the event dispatcher, thus it can watch for events generated by the motor module instead of the device.

Discussion and Future Work

There are some limitations to this model. It does not perform the whole task, and cannot yet drive very far. These limitations suggest changes to JSegMan and its interaction with ACT-R. Specifically, JSegMan should return information about the quality of a match and should use a persistent connection to ACT-R (especially when being used in dynamic environments) to reduce overhead. Finally, JSegMan should incorporate commands that enable models to hold down keys for arbitrary (or indefinite) lengths of time. Implementing these changes will allow JSegMan to be used in modeling more complex tasks. During our work, we also discovered several other interesting topics that can be studied with a model that can drive a *Desert Bus*.

Vigilance

The version of *Desert Bus* we used is multiplayer, allowing other players to enter the bus as passengers. Players can interact with one another by talking or throwing scraps of paper. Thus, cognitive resources are diverted away from driving. Helton and Russel (2011), showed that subjects perform worse at a target detection task when simultaneously performing a spatial or verbal working memory task. Therefore, in the future, the model can be augmented to lose vigilance while driving and interacting with passengers.

Giving Up and Physiologic Effects

Desert Bus is more a game of endurance than skill. The trip, one-way, takes about eight hours to complete and there is no end to the game; the goal is to see how far you can go. A model can play the game forever, but this is unrealistic for a person. A model can be created that weighs external influences and duties against playing and determines when to stop.

Additionally, the model can become more realistic by incorporating physiology with ACT-R/Φ (Dancy, 2013). Players can become hungry, thirsty, and/or sleep deprived while playing, causing their performance to suffer to the point that the bus runs off the road or forces the player to stop. Traditional driving models do not drive for long, so they can ignore these influences. However, ours can theoretically run forever. Adding a physiologic component to the model can reveal interactions between cognition and physiology and leads to a more robust theory of prolonged work and quitting.

Conclusion

With the advent of SegMan and JSegMan, ACT-R gained the capability to truly interact with a wide range of uninstrumented interfaces. ACT-R's motor module has evolved to enable modeling of many behaviors users may exhibit. JSegMan should evolve to make use of the extensions to ACT-R's motor module to allow models to interact with external interfaces with the same behavior as users.

Using *Desert Bus* as a task, we began exploring how to improve JSegMan and what implications our proposals had for modeling and the design of JSegMan in general. While our model did not successfully play the game for long, it yielded useful insights.

Acknowledgements

We would like to thank Jacob Oury and Matt Norris. Oury provided feedback on the paper. Norris found the remake of *Desert Bus*. We also thank everyone who attended the Virtual International Symposium on Cognitive Architectures for providing feedback on the project, particularly Dan Bothell and John Anderson. Work on this project was support by a fellowship to the first author from SMART, and a grant from ONR (N00014-15-1-2275) partially supported the second and third.

References

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* New York, NY: Oxford University Press.

Bothell, D. (2010). Modeling Space Fortress: CMU Effort [PowerPoint slides]. *Seventeenth Annual [ACT-R] Workshop and Summer School.* Retrieved from http://act-r.psy.cmu.edu/wordpress/wp-content/themes/ACT-R/workshops/2010/talks/ICCM_SF.ppt

Byrne, M. D. (2001). ACT-R / PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies*, *55*(1), 41–84. https://doi.org/10.1006/ijhc.2001.0469

Dancy, C.L (2013) ACT-RΦ; A cognitive architecture with physiology and affect. *Biologically Inspired Cognitive Architectures*, *6*(1), 40-45.

Helton, W. S., & Russell, P. N. (2011). Working memory load and the vigilance decrement. *Experimental Brain Research*, 212(3), 429–437. https://doi.org/10.1007/s00221-011-2749-1

Hope, R. M., Schoelles, M. J., & Gray, W. D. (2014). Simplifying the interaction between cognitive models and task environments with the JSON Network Interface. *Behavior Research Methods*, *46*(4), 1007-1012.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer

interaction task. *Transactions on Computer-Human Interaction, 4*(3), 230-275.

Kim, J. W., & Ritter, F. E. (2015). Learning, forgetting, and relearning for keystroke- and mouse-driven tasks: Relearning is important. *Human-Computer Interaction*, *30*(1), 1-33.

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Land, M. F., & Horwood, J. (1995). Which parts of the road guide steering? *Nature, 377*(6547), 339-340.

Land, M. F., & Lee, D. N. (1994). Where we look when we steer. *Nature, 369*(6483), 742-744.

Ritter, F. E., Tehranchi, F., Dancy, C. L., & Kase, S. E. (in press). Some futures for cognitive modeling and architectures: Design patterns that you can too. *Computational and Mathematical Organization Theory*.

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2018). ACT-R: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, *10*(3), e1488.

Schwartz, D. M. & Dancy, C. L. (2019). Building environments for simulation and experimentation in Malmo. *Twenty-Sixth Annual ACT-R Workshop.*

Schwartz, D. M., & Ritter, F. E. (2019). *Lessons from connecting Skirmish Sim and ACT-R/Phi* (Tech. Report No. ACS 2019-3). Applied Cognitive Science Lab, College of Information Sciences and Technology, Penn State.

St. Amant, R., Riedl, M. O., Ritter, F. E., & Reifers, A. (2005). Image processing in cognitive models with SegMan. In *Proceedings of HCI International '05*, Volume 4 - Theories Models and Processes in HCI. Paper # 1869. Mahwah, NJ: Erlbaum.

Tehranchi, F., & Ritter, F. E. (2018a). Modeling visual search in interactive graphic interfaces: Adding visual pattern matching algorithms to ACT-R. *16th International Conference on Cognitive Modeling*, 162–167.

Tehranchi, F., & Ritter, F. E. (2018b). Using Java to provide cognitive models with a universal way to interact with graphic interfaces. *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, Paper LB_15. Washington DC, USA.

Establishing a paradigm to investigate strategy use in complex skills

Roderick Yang Terng Seow (yseow@andrew.cmu.edu)
John R. Anderson (ja0s@andrew.cmu.edu)
Department of Psychology, Carnegie Mellon University, 5000 Forbes Ave
Pittsburgh, PA 15213 USA

Keywords: strategy use; skill acquisition; practice; decision making; adaptivity; ACT-R

Introduction

Questions of strategy selection have been studied in various contexts such as problem solving, text editing, and even dynamic, fast-paced tasks. One way to model the strategy selection process is as a learning and decision problem: with experience, the agent learns the expected utilities of strategies, and executes a strategy based on what it has learned (Lovett & Anderson, 1996).

It is important to note that the strategies studied in most of the past research have relatively stable utilities. Even when the task structure is manipulated to change the utilities of strategies, these changes are relatively infrequent (Schunn & Reder, 2001). This contrasts with many real-world skills, such as sports and video gaming, where different strategies are optimal at different points during the learner's trajectory. As a learner practices a skill, improvements in the learner's degree of perceptual-motor calibration to the physics of tools and devices interacts with the difficulty of executing a strategy to affect the strategy's utility. Furthermore, it is often unknown what the maximum utility of any strategy will be, as this is partly determined by the learner's own general perceptual-motor abilities and prior experiences.

How humans learn and select strategies in the face of such variation and uncertainty behooves further investigation. Towards that goal, we present a task and strategy paradigm that captures many of the features of a typical complex skill. We also examine possible interactions between strategy use, perceptual-motor calibration, and task knowledge using past experimental data and model simulations within the Adaptive Control of Thought-Rational (ACT-R) cognitive architecture.

Space Track

Space Track is a video game developed by Anderson et al. (2019). The player's goal is to earn points in 3-minute trials by navigating a spaceship along a racetrack comprising of rectangular segments (Fig. 1). Completing a rectangle awards 25 points, while crashing into the track walls loses 100 points. Players control the ship using three keys: "A" and "D" to rotate counter-clockwise and clockwise, and "W" to thrust.

Space Track is similar to many sports and video games in three ways. It simultaneously engages visual, motor, and cognitive processing, requires the rapid and precise execution of actions, and has a relatively high performance ceiling. In the study by Anderson et al. (2019), the average human scores around 900 points by the end of 40 trials. This is far under 2350, the highest number of points achieved by one subject.

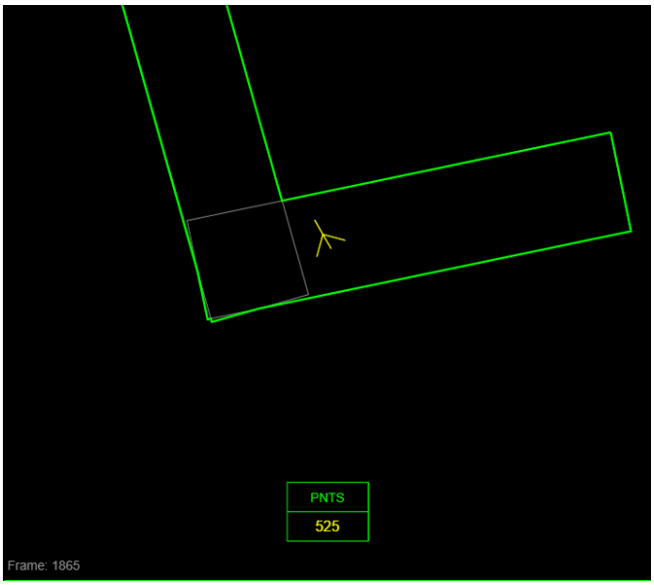


Figure 1: An example Space Track screen

Turning and Calibration

Successful navigation relies on learning the physics of the game. Unlike real-life driving, the racetrack in Space Track is frictionless. To get a car to travel along some desired trajectory, a driver would orient the front wheels to align with that trajectory and then accelerate. If a naïve turner attempts the same in a frictionless space, they will instead send the ship careening away from the desired trajectory due to the residual velocity from the previous trajectory. Turning in frictionless space requires turning less than the desired angle and thrusting until the ship is flying in the desired direction. For any desired speed along the new trajectory there is a unique angle of under-turn that achieves it. Skilled players seem to have learned this angle and how long to thrust.

Through practice, players also gain experience calibrating their perceptual-motor system to account for the continuous and dynamic nature of the game. Human perceptual-motor processing requires time, with an average minimum of 190ms between detecting a visual scene and executing a keypress (Woods et al., 2015). Even a delay of 100 ms between detecting the ship's orientation and lifting the finger from the turn key results in an additional 18 degrees of rotation. Thus, the player needs to account for that lag time by learning the appropriate visual cues for beginning an action. Similarly, a player needs to learn how close to the desired new angle they should be before lifting their thrust finger.

Stopping as a strategy

One potential strategy we identified from prior experiments (Seow et al., 2019) involves rotating the ship in the direction opposite to its current trajectory and thrusting until the ship comes to a halt. Stopping at track corners compensates for naïve turning, since the resultant trajectories from naïve and optimal turners are identical when the ship is stationary. Stopping also gives inexperienced and less calibrated players more time to react to changes in track curvature.

On the flip side, stopping limits average ship speed, which in turn limits the maximum possible distance a player can cover within each 3-minute trial. Thus, it is not immediately obvious whether stopping has a positive or negative utility.

Using data from Seow et al. (2019), we found that increased stopping did not predict a change in the average points earned per trial (Fig 2). Rather, increased stopping correlated with a narrower range of points, raising the floor while lowering the ceiling. This suggests that when scoring is below the mean, increased stopping use might improve performance, but when it is above that mean, increased stopping use might instead lead to worse performance.

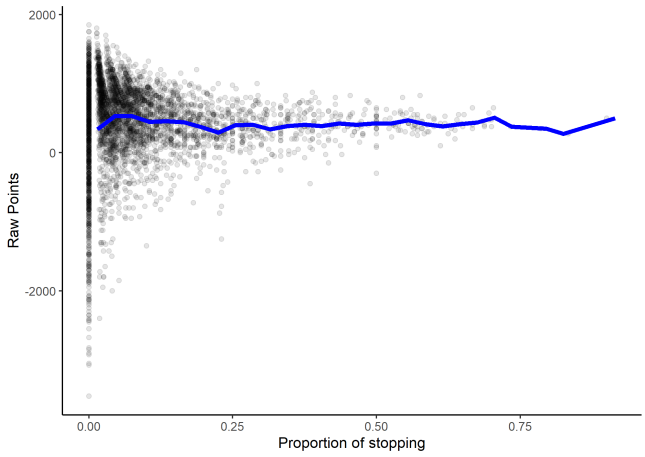


Figure 2: Points per trial in humans. The blue line tracks the average points earned across proportions of stopping.

Stopping, Turn Optimality, and Experience

To test these potential tradeoffs of stopping, we simulated learning on Space Track using ACT-R models. We modeled variations in turn optimality as 11 weighted combinations of the contributions from the naïve and optimal turning algorithms. Differences in stopping use was captured by two types of models, one that stopped at every corner, and one that relied on its turn algorithm to navigate corners. Stopping use was crossed with turn optimality to yield 22 models.

Stopping helps inexperienced agents and naïve turners but limits experienced agents and optimal turners (Fig. 3). A regression model ($r^2 = 0.74$) further showed that score was predicted by the interaction between stopping, turn optimality, and trial number ($\beta = 2.19$, $SE = 0.65$, $p < 0.05$).

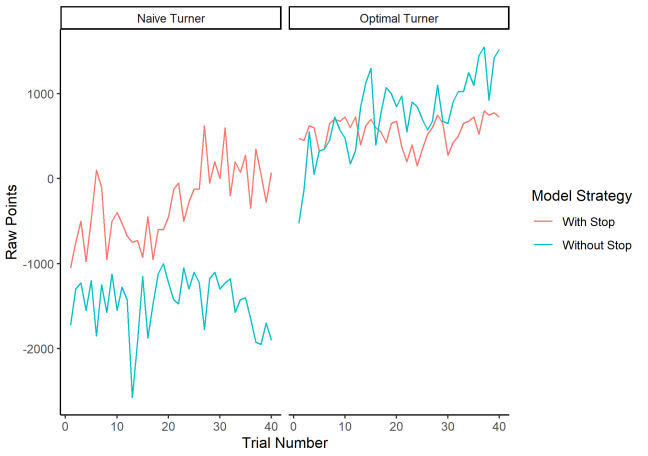


Figure 3: Model points across trials. The stopping model (in red) performs better than the non-stopper except when the agent is an optimal turner and has had sufficient experience.

Conclusion and Further Research

We have identified a task and strategy paradigm that is potentially suitable for understanding the processes that underlie the learning and selection of strategies in complex skills. With Space Track and the strategy of stopping, one promising future direction is to apply and test current models of decision making (e.g. Reinforcement Learning) on human gameplay to investigate how basic decision processes account for strategy shifts in complex skill acquisition.

Acknowledgments

This research was supported by ONR Grant N00014-15-1-2151 and AFOSR/AFRL award FA9550-18-1-0251.

References

Anderson, J. R., Betts, S., Bothell, D., Hope, R., & Lebiere, C. (2019). Learning rapid and precise skills. *Psychological review*, *126*(5), 727.

Lovett, M. C., & Anderson, J. R. (1996). History of success and current context in problem solving: Combined influences on operator selection. *Cognitive psychology*, *31*(2), 168–217.

Schunn, C. D., & Reder, L. M. (2001). Another source of individual differences: Strategy adaptivity to changing rates of success. *Journal of experimental psychology: General*, *130*(1), 59.

Seow, R., Betts, S., & Anderson, J. R. (2019). Transfer effects of varied practice and adaptation to changes in complex skill acquisition. In *Proceedings of the 17th international conference on cognitive modelling* (p. 222-227).

Woods, D. L., Wyma, J. M., Yund, E. W., Herron, T. J., & Reed, B. (2015). Factors influencing the latency of simple reaction time. *Frontiers in human neuroscience*, *9*, 131.

Modeling the Absence of Framing Effect in an Experience-based Covid-19 Disease Problem

Neha Sharma (neha724@gmail.com)
School of Computing, Indian Institute of Information Technology Una, India – 177220

Shashank Uttrani (shashankuttrani@gmail.com)
Applied Cognitive Sciences Laboratory, Indian Institute of Technology Mandi, India – 175075

Varun Dutt (varun@iitmandi.ac.in)
Applied Cognitive Sciences Laboratory, Indian Institute of Technology Mandi, India – 175075

Abstract

Prior research in decisions from experience (DFE) has investigated people's consequential decisions after information search both experimentally and computationally. However, prior DFE research has yet to explore how computational cognitive models and their mechanisms could explain the effects of problem framing in experience. The primary objective of this paper is to address this literature gap and develop Instance-based Learning Theory (IBLT) models on the effects of problem framing. Human data was collected on a modified form of the Asian disease problem posed about the COVID-19 pandemic across two between-subject conditions: gain (N = 40) and loss (N = 40). The COVID-19 problem was presented as "lives saved" in the gain condition and "lives lost" in the loss condition. Results revealed the absence of the classical framing effect, exhibiting no preference reversal between gain and loss conditions in experience. Next, an IBL model was developed and calibrated to the data obtained in the gain and loss problems. The calibrated model was generalized to the non-calibrated conditions (gain to loss and loss to gain). An IBL model with ACT-R default parameters was also generalized. Results revealed that the IBL model with calibrated parameters explained human choices more accurately compared to the IBL model with ACT-R default parameters. Also, participants showed greater reliance on recency and frequency of outcomes and less variability in their choices across both gain and loss conditions. We highlight the main implications of our findings for the cognitive modeling community.

Keywords: individual choice; experience; sampling; computational models; framing; gain; loss; COVID-19 disease problem.

Introduction

Whenever the world has seen new contagious diseases, medical practitioners have relied on their prior experience with treatments on other diseases to tackle the crises (HT, 2020). Depending upon the similarity and differences between prior experiences of diseases and a specific current disease, a combat plan may be selected for implementation. The act of making choices based upon prior experience, however, is not limited to making disease combat decisions; rather, it may be a very common exercise involving people in different facets of their daily life (choosing what to eat, whom to marry, or what career to pursue). Gaining experience via information search (or sampling) before a consequential choice forms an integral part of decisions from experience (DFE) research, where the focus is on explaining human decisions based upon one's experience with sampled information (Hertwig & Erev, 2009).

DFE research has proposed a "sampling paradigm" (Hertwig & Erev, 2009), where people are presented with two or more options to choose between. These options are represented as blank buttons on a computer screen. People can sample as many buttons as they wish and in any order they desire (information search). Once people are satisfied with their sampling of the button options, they decide from which option to make a single consequential choice for real.

The sampling paradigm has been used to develop computational cognitive models of human choice behavior both at the individual level (Sharma & Dutt, 2017) and at the aggregate level (Busemeyer & Wang, 2000; Gonzalez & Dutt, 2012; Lejarraga, Dutt, & Gonzalez, 2012). In fact, using the sampling paradigm, cognitive models have also been developed in both abstract and applied domains (Sharma & Dutt, 2018). For example, the Instance-Based Learning (IBL) model is a popular DFE algorithm for explaining aggregate and individual human choices (Erev et al., 2010; Gonzalez & Dutt, 2011; Sharma & Dutt, 2017). The IBL model borrows mechanisms like activations, retrieval from memory, and blending from the ACT-R framework (Anderson & Lebiere, 1998) and it operates by storing and retrieving experiences (called instances) from memory (Gonzalez & Dutt, 2011). Each instance's activation is used to calculate the blended values for each option, thereby helping the model to make a consequential choices.

Although computational cognitive models have been developed in the DFE's sampling paradigm at the aggregate and individual participant levels in abstract and applied problem domains (Sharma & Dutt, 2017), yet little is known on how these models would account for human decisions driven by the problem's framing in experience in applied domains (Gonzalez, Dana, Koshino, & Just, 2005; Tversky & Kahneman, 1981). For example, in the famous Asian disease problem (ADP), participants are asked to imagine that a country is preparing for the outbreak of an unusual Asian disease, which is expected to kill 600 people (Tversky & Kahneman, 1981). One group of people are presented this problem as a gain in terms of "lives saved;" whereas, a second group of people are presented the same problem as a

loss in terms of "lives lost." Although the gain and loss frames are equivalent, results reveal a framing effect: A large majority among those presented the gain frame choose the safe option; however, a large majority among those presented the loss frame choose the risky option. Gonzalez and Mehlhorn (2015) showed that the framing effect was present among people when they were presented with the ADP in a descriptive format; however, the framing effect disappeared in the experiential format (i.e., DFE's sampling paradigm). Gonzalez and Mehlhorn (2015) went a step further and developed an IBL model with ACT-R parameters to explain the disappearance of the framing effect in experience. However, Gonzalez and Mehlhorn (2015) did not calibrate their model's parameters as well as these authors did not test the framing effect in problems with a context (e.g., problem about the specific COVID-19 disease compared to the general Asian disease).

The primary objective of this research is to overcome the above-mentioned literature gaps. First, we evaluate the framing effect in experience among gain and loss problem frames in a COVID-19 disease problem (CDP). Next, we evaluate how an IBL model calibrated to the gain and loss frames explains the human choices in CDP. We also evaluate the generalization of IBL model parameters from the calibrated problem to the non-calibrated problem (gain problem's parameters to the loss problem and loss problem's parameters to the gain problem). For the purposes of our evaluations, the IBL model was exposed the sampling of participants and it predicted the consequential choices post sampling.

In what follows, first, we detail an experiment where we investigated the framing effect in experience in CDP. Next, we detail an IBL model and discuss the methodology of calibrating the model to capture the consequential choices in CDP. Next, we present the results of model's evaluation both during calibration and during generalization. Finally, we close the paper by discussing the implications of our results.

The COVID-19 Disease Problem (CDP) Experiment

Eighty participants were recruited via Amazon MTurk in India to participate in a disease program study. Participation was voluntary, about 67% percent of participants were males, and the rest were females. Ages ranged from 18 years to 73 years (Mean = 32.55 years and standard deviation = 10.04 years). Participants were from different education levels: 19.4% undergraduates and 80.6% graduates. Discipline-wise, the demographics were the following: 31.25% possessed degrees in engineering, 10.62% possessed degrees in basic sciences, and 28.6% possessed degrees in humanities and social sciences. Participants were compensated a flat participation fee of INR 21 (~ USD 0.28). No participant took more than 10 minutes across both conditions to finish the study.

Participants were randomly assigned to one of two between-subject conditions involving the CDP in experience: gain (N = 40) and loss (N = 40). In the gain condition, the CDP was framed as "lives saved;" whereas, in the loss condition, the CDP was framed as a "lives lost" (see Figure 1).

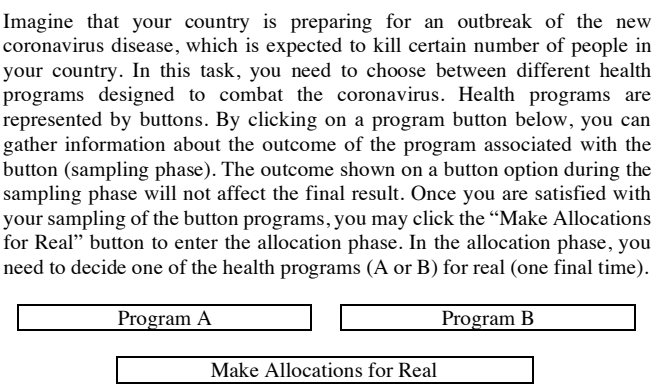


Figure 1. The CDP presented to participants in the study in gain condition.

As shown in Figure 1, in the gain condition, participants were presented with programs A and B, which they needed to sample as many times they desired and in any order they desired before making a final choice for real. In the loss condition, participants were presented with programs C and D, which they needed to sample as many times they desired and in any order they desired before making a final choice for real. The allocation of programs to buttons was randomized across participants in both conditions and sampling in both conditions was nonconsequential. At any time during the sampling phase, participants could click the "Make Allocations for Real" button (see Figure 1). Clicking the "Make Allocations for Real" button terminated the sampling phase and moved participants to the allocation phase. In the allocation phase, participants were asked to make a consequential choice for one of the programs. In the gain condition, program A was framed as "200 people will be saved" (1 probability) and program B was framed as "600 people will be saved" (1/3rd probability) or "No one will be saved" (2/3rd probability). In the loss condition, program C was framed as "400 people will die" (1 probability) and program D was framed as "Nobody will die" (1/3rd probability) or "600 will die" (2/3rd probability). In both conditions, the probability information was not shown, and it was only used to generate the outcomes in quotes above. As can be seen, programs A and C were identical and programs B and D were identical. In agreement with Gonzalez and Mehlhorn (2015)'s results for ADP, we expected no difference in the proportion of A and C choices in the CDP (i.e., we expected an absence of the framing effect). To test our expectation, we performed a one-way ANOVA with condition as a between-subjects factor, an alpha level of 0.05, and a power of 0.80.

Results revealed that there was no significant difference between the gain and loss conditions in the proportion of A

or C choices (gain: 0.83 ~ loss: 0.70; $F(2,78) = 1.720$, $p = .19$, $\eta^2 = 0.02$). Thus, as per our expectations and contrary to the classical descriptive results, there was an absence of the framing effect in the experience-based CDP.

The Model

In this section, we detail the working of the IBL model that was developed to account for human choices in the CDP.

Instance-Based Learning (IBL) Model

The IBL model (Dutt & Gonzalez, 2012; Gonzalez & Dutt, 2011; 2012; Lejarraga, Dutt, & Gonzalez, 2012) is built upon the ACT-R cognitive framework (Anderson & Lebiere, 1998). In this model, instances are created in memory for each occurrence of an outcome on choice options. An instance is made up of the following structure: situation-decision-utility, where the situation is the current situation (two option buttons on a computer screen), the decision is the decision made in the current situation (choice for one of the option buttons), and the utility is the goodness of the made decision (the outcome obtained upon choosing an option). When a choice is to be made, instances belonging to each option are retrieved from memory. These instances are then blended on each option. The blended value of an option is a function of activation of instances as well as their probability of retrieval from memory. The blended value of option j at any trial t is defined as:

$$V_{j,t} = \sum_{i=1}^{n} p_{i,j,t} x_{i,j,t} \qquad (1)$$

where $x_{i,j,t}$ is the value of the utility part of an instance i on option j at trial t. The $p_{i,j,t}$ is the probability of retrieval of instance i on option j from memory at trial t. Because $x_{i,j,t}$ is the utility of an instance i on option j at trial t, the number of terms (n) in the summation in equation 1 changes when new outcomes are observed during sampling on the option j. For example, if j is an option with two possible outcomes, then n = 1 when one of the outcomes has been observed on the option (i.e., one instance is created in memory) and n=2 when both outcomes have been observed on the option (i.e., two instances are created in memory).

At any trial t, the probability of retrieval of an instance i on option j at trial t is a function of the activation of that instance relative to the activation of all instances (1, 2, … n) created within the option j, given by

$$p_{i,j,t} = \frac{e^{(A_{i,j,t})/\tau}}{\sum_{i=1}^{n} e^{(A_{i,j,t})/\tau}} \qquad (2)$$

where τ, is random noise defined as $\sigma * \sqrt{2}$ and σ is a free cognitive noise parameter. The activation of an instance i corresponding to an observed outcome on an option j in a given trial t is a function of the frequency of the outcome's past occurrences and the recency of the outcome's past occurrences (as done in ACT-R). At each trial t, activation $A_{i,j,t}$ of an instance i on option j is

$$A_{i,j,t} = \sigma * \ln\left(\frac{1-\gamma_{i,j,t}}{\gamma_{i,j,t}}\right) + \ln \sum_{t_p \varepsilon \{1..,t-1\}} \left(t - t_p\right)^{-d} \qquad (3)$$

where d is a free decay parameter; $\gamma_{i,j,t}$ is a random draw from a uniform distribution bounded between 0 and 1, for instance i on option j in trial t; and t_p is each of the previous trials in which the outcome corresponding to instance i was observed in the task. The IBL model has two free parameters that need to be calibrated: d and σ. The d parameter controls the reliance on recent or distant sampled information. Thus, when d is large (> 1.0), then the model gives more weight to recently observed outcomes in computing instance activations compared to when d is small (< 1.0). The σ parameter helps to account for the participant-to-participant variability in an instance's activation. We feed the sampling done by individual human participants to generate instances and compute blended values in the IBL model. During sampling, each time a choice is made, and the outcome is observed, the instance associated with it is activated (created or reinforced). At the final choice, blended values are computed and the model chooses the option with the highest blended value.

In one version of the IBL model, we used the default values of the ACT-R parameters, i.e., $d = 0.50$ and $\sigma = 0.25$ (IBL model with ACT-R parameters). These parameters show lesser reliance on recency and frequency of information and a reasonable participant-to-participant variability in consequential choices. However, in a second version of the IBL model, we found single values for the two parameters (d and σ) by calibrating them to individual participant consequential choices in gain and loss conditions, respectively. We refer to this model as the IBL model with calibrated parameters and, for the parameters' calibration, we determined a model participant's choice and compared this choice to a human participant's choice. In order to create exploration of options during sampling, the model's memory was pre-populated with 2 instances (i.e., one on each option) with a 1000 utility. This value of utility was higher than all possible outcomes in the different options. These prepopulated instances may represent the initial expectations that participants may bring to the task (Gonzalez & Dutt, 2011). If the model participant's choice equaled human participant's choice, then the dependent variable (error) was coded as zero; otherwise, the error was coded as one. We minimized the average of errors across all participants in the calibration process separately across the gain and loss conditions.

Method

Dependent Variables

The model was run for as many model participants as there were human participants in the two conditions independently.

To compare human and model choices, we evaluated an "error ratio" (i.e., the ratio of incorrectly classified final choices between model and human participants divided by the total number of human participants). Thus, the error ratio was calculated as:

$$\text{Error Ratio} = (A_mB_h + B_mA_h) / (A_mA_h + B_mB_h + A_mB_h + B_mA_h) \quad (4)$$

where, A_mB_h was the number of participants where the model predicted an A (or C) program choice but the human made a B (or D) program choice. B_mA_h was the number of participants where the model predicted a B (or D) program choice but the human player made a A (or C) program choice. Similarly, the A_mA_h and B_mB_h were the number of participants, where the model predicted the same choice as made by the human participant. The smaller the value of the error ratio, the more accurate is the model in accounting for individual choices in CDP.

Model Calibration

The IBL model described here possessed two free parameters *d* and σ. These parameters were calibrated using a genetic algorithm program in both gain and loss conditions separately. The genetic algorithm repeatedly modified a population of individual parameter tuples in order to find the tuple that minimized the error ratio in a condition. The *d* and σ parameters were both varied in the range [0, 10]. In each generation, the genetic algorithm selected individual parameter tuples randomly from a population to become parents and used these parents to select children for the next generation. Over successive generations, the population evolved toward an optimal solution. The population size used here was a set of 20 randomly selected parameter tuples in a generation (each parameter tuple was a particular value of *d* and σ). The mutation and crossover fractions were set at 0.1 and 0.8, respectively, for an optimization over 150 generations. For each parameter tuple, the IBL model was run 10 times across the 40 human participants per condition to account for run-to-run uncertainties present in the model. Across the 10 runs, the model's average error ratio was computed by averaging the error ratios from each run and it was minimized. The parameter tuple that minimized the average error ratio across 150 generations were reported as the calibrated parameters for the IBL model.

Results

We evaluated the IBL model's ability to account for individual consequential choices in both gain and loss conditions separately. In the gain condition, the best-calibrated values of *d* and σ parameters were found to be 7.05 and 0.07, respectively. In the loss condition, the best-calibrated values of *d* and σ were found to be 9.70 and 0.22, respectively. A large *d* value exhibited excessive reliance on recency during sampling. Also, the smaller σ value exhibited lesser participant-to-participant variability in instance activations.

Table 1 shows the individual-level results from the gain and loss conditions. The same results were obtained across 10-runs of the model and there was no deviation in the percentages from the mean. As shown in Table 1, the calibrated IBL model in the gain condition produced 83% of A_mA_h combinations and 17% of B_mB_h combinations, respectively. In contrast, for the IBL model in the gain condition, the erroneous A_mB_h and B_mA_h combinations were both 0.0%, respectively. Based on these statistics, the IBL model showed 100% accuracy in the gain condition. Furthermore, the calibrated IBL model in the loss condition produced 70% of A_mA_h combinations and 30% of B_mB_h combinations, respectively. In contrast, for the IBL model in the loss condition, the erroneous A_mB_h and B_mA_h combinations were both 0.0%, respectively. Thus, again, the IBL model possessed 100% accuracy in the loss condition.

Table 1: The calibration results from the IBL model in the CDP.

Human and Model data combination H/M	Gain condition	Loss condition
Parameters	$d = 7.05$, $\sigma = 0.06$	$d = 9.70$, $\sigma = 0.22$
Number of participants	40[1]	40
A_mA_h percentage	83	70
B_mB_h percentage	17	30
A_mB_h percentage	00	00
B_mA_h percentage	00	00
Error Ratio	00	00

Note. [1] Each of the 10-runs of the model produced the same percentage with 0.0 as the standard deviation.

Table 2 shows the results of the IBL model in the CDP where the model possessed ACT-R default parameters ($d = 0.5$ and $\sigma = 0.25$).

Table 2: The IBL model in the CDP with ACT-R default parameters.

Human and Model data combination H/M	Gain condition	Loss condition
Parameters	$d = 0.50$, $\sigma = 0.25$	$d = 0.50$, $\sigma = 0.25$
Number of participants	40	40
A_mA_h percentage	41.6[1] (5.5)[2]	34.5 (5.9)
B_mB_h percentage	14.7 (3.0)	13.0 (2.6)
A_mB_h percentage	03.3 (3.0)	17.0 (2.6)
B_mA_h percentage	41.4 (5.5)	35.5 (5.9)
Error Ratio	0.45 (0.10)	0.53 (0.10)

Note. [1] The average percentage across 10-runs. [2] The standard deviation across 10-runs.

As seen in Table 2, in the gain condition, there were, on average, 41.6% of A_mA_h combinations and 14.7% of B_mB_h combinations, respectively. In contrast, on average, the erroneous A_mB_h and B_mA_h combinations were 3.3% and 41.4%, respectively. The average error ratio being 0.45. In the loss condition, on average, there were 34.5% of A_mA_h combinations and 13.0% of B_mB_h combinations, respectively.

In contrast, on average, the erroneous A_mB_h and B_mA_h combinations were 17.0% and 35.5%, respectively. The average error ratio being 0.53. Overall, the IBL model with ACT-R default parameters performed poorly compared to the calibrated IBL model.

Figure 2 shows the proportion of A choice (gain condition) or proportion of C choices (loss condition) from human data, IBL model with calibrated parameters, and IBL model with ACT-R default parameters. As can be seen in the Figure, the IBL model with calibrated parameters captured the human choices accurately; whereas, the IBL model with the ACT-R default parameters exhibited a close to a chance performance.

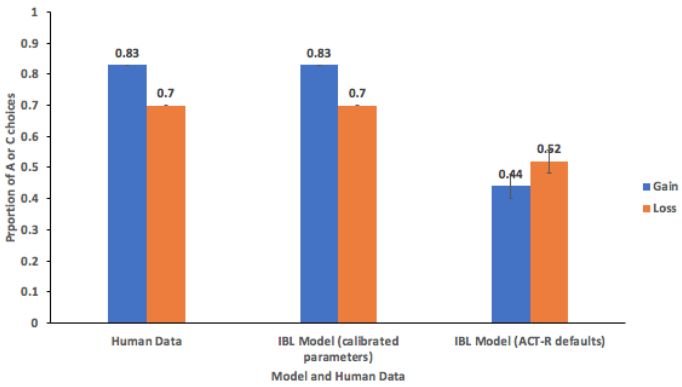


Figure 2. The proportion of A choice (gain condition) or C choices (loss condition) in human data, calibrated IBL model, and IBL model with ACT-R default parameters. The error bars show 95% CI around the average estimate.

Generalization

Since, we first calibrated the IBL model in the gain and loss conditions independently, generalizing the model by running the calibrated parameters in the non-calibrated conditions (from loss condition to gain condition or from gain condition to loss condition) would help account for parameter differences and model consistency across the two conditions. As there was an absence of the framing effect in the experimental data, generalization of loss condition parameters to the gain condition or generalization of gain condition parameters to the loss condition should produce accurate and similar results.

Table 3 shows the results of generalizing the IBL model from calibrated conditions to the non-calibrated conditions. The same results were obtained across 10-runs of the model and there was no deviation in the percentages from the mean. As shown in Table 3, the generalization of loss condition's parameters in the gain condition and the generalization of gain condition's parameters in the loss condition produced most accurate results with 0 error ratios. Thus, these parameters are equivalent and they meet our expectations on the absence of the framing effect in the experienced-based CDP.

Table 3: Generalisation of the calibrated IBL model parameters from calibrated condition to the non-calibrated conditions in CDP.

Human and Model data combination H/M	Loss condition's parameters in gain condition	Gain condition's parameters in loss condition
Parameters	$d = 9.70$, $\sigma = 0.22$	$d = 7.05$, $\sigma = 0.06$
Number of participants	40[1]	40
A_mA_h percentage	83	70
B_mB_h percentage	17	30
A_mB_h percentage	00	00
B_mA_h percentage	00	00
Error Ratio	00	00

Note. [1] Each of the 10-runs of the model produced the same percentage with 0.0 as the standard deviation.

Discussion and Conclusions

Prior research had experimented with the framing effect in the Asian disease problem (ADP) where the problem was presented in gain and loss frames either in a descriptive format (description) or experiential format (experience) to participants (Gonzalez et al., 2005; Gonzalez and Mehlhorn 2015; Tversky & Kahneman, 1981). The main result was the presence of the framing effect (i.e., a preference reversal) between gain and loss problems in description and its absence in experience. However, little was known about the existence of the framing effect in problems with an applied disease context (e.g., COVID-19) in experience. Also, little was known about how computational cognitive models could account for the framing effect in applied disease contexts in experience. The primary objective of this research was to address these gaps in literature. In this paper, we showed the absence of the framing effect between the gain and loss frames in an applied COVID-19 disease problem (CDP) in experience. Furthermore, we showed that a single IBL model could account for the absence of framing effect in both gain and loss frames in the CDP in experience. The IBL model showed participants relying excessively on recency and frequency of information and showing very little variability in participant-to-participant decisions across both the gain and loss frames in CDP.

First, our experimental results showed an absence of the framing effect across the gain and loss frames in CDP in experience. This result is consistent with those of Gonzalez and Mehlhorn (2015), who also showed the absence of the framing effect across the gain and loss frames in the experience-based ADP. Thus, it seems that the specific COVID-19 disease context (in CDP) is treated by participants in the same manner as the general Asian disease context (in ADP). One likely reason for the absence of the framing effect between gain and loss frames in CDP could be that in experience, people underweight the probability of low frequency events and overweight the probability of the high frequency events (Hertwig & Erev, 2009). Here, this effect of underweighting and overweighting of probabilities seems to be present irrespective of the problem's framing.

Second, our model results showed that the IBL model with calibrated parameters performed exceedingly well compared to the IBL model with ACT-R default parameters. This result extends the work of Gonzalez and Mehlhorn (2015), who developed an IBL model with the ACT-R default parameters. Specifically, it shows that in experience, the default ACT-R assumptions of low recency and reasonable variability may not exist, and people may make more deterministic final choices that are driven by excessive reliance on recent and frequent samples.

Third, our results showed that the generalization of model parameters from calibrated conditions to non-calibrated conditions showed very accurate model performance. This results in particular shows that the IBL model parameters in the gain and loss frames were similar and these parameters tended to agree with the experimental findings on the absence of the framing effect: If there is no preference reversal between gain and loss frames, then the model parameters should also similarity in their values.

There are a number of future directions from this work. First, researchers may also develop the CDP in description and experimentally evaluate whether there is a presence of the framing effect in the description-based CDP. Next, researchers may attempt whether there is an effect of the people's location (being in Asia or in America) on the contextual disease framing in experience and description formats. Furthermore, cognitive models like IBL may be developed on these data to see the potential of such models in capturing the presence or absence of framing effects. In this paper, only base level activation as well as the cognitive noise were used in explaining the framing effect in the IBL model. However, future work may experiment with other ACT-R mechanisms like partial matching or spreading activation to account for these experimental findings. We plan to continue experimenting with some of these ideas as part of our future work in the decisions from experience theme.

Acknowledgments

The authors are grateful to the Indian Institute of Information Technology Una, and Indian Institute of Technology Mandi for providing computational resources for this paper.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates. ISBN 0-8058-2817-6

Busemeyer, J. R., & Wang, Y. (2000).Model comparisons and model selections based on the generalization criterion methodology. *Journal of Mathematical Psychology*, 44, 171–189.

Brandstätter, E., Gigerenzer, G., & Hertwig, R. (2006). The priority heuristic: making choices without trade-offs. *Psychological review*, *113*(2), 409.

Dutt, V. and Gonzalez, C. (2012). The role of inertia in modeling decisions from experience with instance-based learning. *Frontiers in Psychology* 3:177. DOI: 10.3389/fpsyg.2012.00177

Erev, I., Glozman, I., & Hertwig, R. (2008). What impacts the impact of rare events. *Journal of Risk Uncertainty* 36:153–177.

Erev, I., Ert, E., Roth, A. E., Haruvy, E., Herzog, S. M., & Hau, R.(2010). A choice prediction competition: Choices from experience and from description. *Journal of Behavioral Decision Mak*ing, 23, 15–47.

Everitt, B. S. (1998). Dictionary of statistics.

Gonzalez, C., Dana, J., Koshino, H., & Just, M. (2005). The framing effect and risky decisions: Examining cognitive functions with fMRI. *Journal of economic psychology*, *26*(1), 1-20.

Gonzalez, C.,& Dutt, V. (2012). Refuting data aggregation arguments and how the instance-based learning model stands criticism: A reply to Hills and Hertwig *Psychological Review*, *Vol* 119(4),893-898.

Gonzalez, C., & Dutt, V. (2011). Instance-Based Learning: Integrating Sampling and Repeated Decisions From Experience. *Psychological Review*,118(4), 523-551.

Gonzalez, C., & Mehlhorn, K. (2015). Framing From Experience: Cognitive Processes and Predictions of Risky Choice. *Cognitive Science*, *1*, 29. DOI: 10.1111/cogs.12268

Hertwig, R., & Erev, I. (2009). The description-experience gap in risky choice. *Trends in Cognitive Sciences*, 13, 517-523.

Hertwig, R., & Pleskac, T. J. (2010). Decisions from experience: Why small samples*?*. *Cognition*, 115,225–237.

Hertwig, R. (2012). The psychology and rationality of decisions from experience. *Synthese*, 187, 269-292.

Lebiere, C. (1999). Blending: An ACT–R mechanism for Aggregate retrievals. *Paper presented at the 6th Annual ACT–R Workshop at George Mason University*. Fairfax County, VA.

Lejarraga, T. & Dutt, V. & Gonzalez, C. (2012). Instance-Based Learning: A general model of repeated binary choice. *Journal of Behavioral Decision Making*,25: 143-153.

PGI to test leprosy vaccine on Covid-19 patients(2020, Apr 22) Retrieved from https://www.hindustantimes.com/chandigarh/pgi-to-test-leprosy-vaccine-on-covid-19-patients/story-0DHH4GS8mANbUZNRZj2kYK.html

Rieskamp, J. (2008). The probabilistic nature of preferential choice. Journal of Experimental Psychology: Learning, Memory, and Cognition, 34(6), 1446.

Sharma, N., & Dutt, V. (2017). Modeling decisions from experience: How models with a set of parameters for aggregate choices explain individual choices. *Journal of Dynamic Decision Making*, *3*, 3-3.

Sharma, N., Debnath, S., & Dutt, V. (2018). Influence of an Intermediate Option on the Description-Experience Gap

and Information Search. *Frontiers in psychology*, *9*, 364.
Tversky, A., & Kahneman, D. (1981). The framing of decisions and the psychology ofchoice. *science*, *211*(4481), 453-458.
Tversky, A., & Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty*,*5* (4), 297-323.

The Need for Speed: Effects of Human Derived Time Constraints on Performance and Strategy in Machine Models of Tetris

Catherine Sibert (sibert@uw.edu)
Department of Psychology, Campus Box 351525
Seattle, WA 98195 USA

Wayne D. Gray (grayw@rpi.edu)
Cognitive Science, Rensselear Polytechnic Institute, 110 8th Street
Troy, NY 12180 USA

Abstract

One of the hallmarks of expert performance in complex, dynamic tasks is the ability to select and perform the appropriate action within a constantly shifting environment, often under tight time constraints. In an example task, the video game Tetris, expert players select placement positions for the active zoid and navigate them into place in increasingly short time spans. Machine models of the same task are capable of producing human-like performance patterns, but either ignore or only roughly approximate the time constraints that seem to be an integral part of human behavior. Using a set of scaled time parameters derived from a large set of human players, we trained and tested an existing machine Tetris model and observed the resultant changes in performance and behavior.

Keywords: Expertise, Reinforcement Learning, Machine Learning, Human Performance

Introduction

Expertise is marked by the ability to perform a particular task at a very high level of proficiency, and in many task domains, are capable of performing more quickly and efficiently than non experts. But while speed is often observed in conjunction with high levels of skill, it is not always clear how it contributes to performance. In this paper, we explore the relationship between speed and strategy in a complex and dynamic task environment, the video game Tetris.

Video games, and Tetris in particular, have a long history of use in research. Gray (2017) identifies three major uses of games: *Gamification* describes efforts to use a game-like environment for more serious and real world applications (Rapp, Cena, Gena, Marcengo, & Console,2016;Nash & Shaffer,2011;Proctor, Bauer, & Lucario,2007), games as *Treatment Conditions* are studies that use games to alter some aspect of human behavior (Holmes, James, Coode-Bate, & Deeprose,2009;Belchior et al.,2013), and *Game-XP* is a term referring to the use of a game as an experimental paradigm. in this work we use Tetris as a task environment to investigate the low level mechanisms that give rise to high level skilled behavior (Kirsh & Maglio,1994;Destefano, Lindstedt, & Gray,2011;Lindstedt & Gray,2019;Sibert, Gray, & Lindstedt,2017;Sibert & Gray,2018). Exploring this task could help to us understand how these skills are developed and how complex strategies are learned and used.

Tetris the Task

Tetris is a real-time, dynamic puzzle solving game that is simple in concept and can be very complex in execution. A player is presented with a sequence of game pieces, called “zoids”, made up of four conjoined squares. These zoids fall from the top of the game board, and as they fall, the player navigates them into position using a series of translation and rotation maneuvers. Placed zoids form a pile at the bottom of the screen, and when a row of the pile is completely filled across the width of the screen, that row will disappear, lowering the pile and earning the player some points. The game ends if the pile reaches the top of the screen. Higher scores are achieved by clearing multiple lines (up to four) simultaneously, which encourages players to plan ahead and construct board structures that can support these more complex maneuvers. However, as lines are cleared, the zoid falling speed increases, allowing players less and less time to plan and execute their moves. At early game levels, players have a full 16 seconds for maneuvering, but that time window is slowly reduced until the fall speed of a zoid is a third of a second at the highest playable levels. Successful players must balance the score benefits of complex multi-line clears with the risk of building the pile too high and having insufficient time and space to maneuver zoids into place.

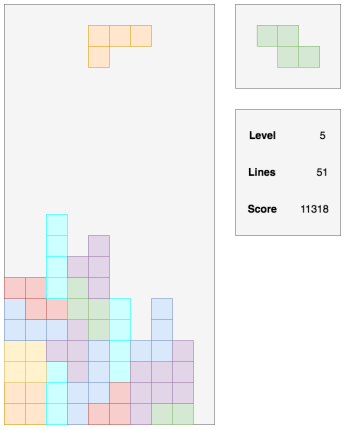


Figure 1: A Tetris game in progress. The active piece, the orange ”L” is currently being placed by the player on the main game screen. The player also has access to score information, in the lower right-hand box, and one upcoming piece, the green ”Z” in the upper right-hand box.

The constantly changing nature of Tetris allows for a wide

range of player skill, with extremely low level players struggling to clear any lines at all, all the way up to extreme experts, who comfortably set up and execute multi-line clears in fractions of a second. Performance in this task is usually judged by the final game score, but that score is achieved though a complex interaction of perceptual, cognitive, and motor skills.

Tetris Models

Building models of Tetris has long been pursed in the Machine Modeling community (Robertson,2003;Fahey,2015;Szita & Lorincz,2006;Thiery & Scherrer,2009a), and these efforts have produced computer models capable of high level Tetris performance. These models are not usually developed out of any specific interest in Tetris behavior, but they use Tetris as a testbed to demonstrate the effectiveness of machine learning algorithms that optimize a large feature space.

Though the search methods and models are all different across the machine learning work, the basic model structure is fairly consistent.The researcher selects a set of features of interest, often structural aspects of the board like the pile height, or number of unfilled cells, that may be important when making placement decisions. Each of these features is assigned a numerical weight, and using these weights, the model is able to calculate a numerical score for all potential placements for any static board state. A game is played by simply selecting the highest scoring move at each decision point, and updating the game board to reflect the previous decision. A simple, yet effective, set of features was defined by Dellacherie (Fahey,2015) and used in the modeling experiments presented here. The features are described in Table 1.

Table 1: Tetris features proposed by Dellacherie, and used to construct the models used in this paper

Feature	Description
Landing Height	Height where the last zoid is added
Eroded Cells	# of cells of the current zoid eliminated due to line clears
Row Transitions	# of full to empty or empty to full horizontal transitions between cells on the board
Column Transitions	# of full to empty or empty to full vertical transitions between cells on the board
Pits	# of empty cells covered by at least one full cell
Wells	a series of empty cells in a column such that the cells to the left and right are both full

As suggested by Table 1, most machine modeling work on Tetris has focused on selecting weights for a limited set of features. These weights must be able to select moves that result in high game scores, but must do so in a nearly infinitely variable environment of potential board configurations. A machine learning method that is capable of producing a successful model of Tetris, then, is likely to also be effective in other complex task domains. One such method, and the one employed in this research, is *Cross Entropy Reinforcement Learning* (CERL), first proposed by Szita and Lorincz (2006) and modified by Thiery and Scherrer (2009a,2009b), and uses a generational search method to narrow in on the optimal feature space.

The models produced by this line of research are very effective Tetris players, clearing hundreds of thousands of lines in a game (a high scoring human clears five or six hundred), but they do so by adopting very un-human-like strategies that allow them to take advantage of significant differences between the human and model task environment. Most notably, the models are completely unconstrained by time pressure. Where human strategy often revolves around making and executing the best placement decisions in the time available, the models instantly choose which of the possible zoid placements has the highest rank (the highest number of alternative placements possible for one episode is 35). For a model player, the game ends when the feature weights encounter a sequence of zoids that it cannot place in a way that clears any lines, far different from the time constraints that limit human players. In response, models and humans develop divergent strategies. The model player emphasizes clearing single lines repeatedly over very long game spans (behavior also exhibited by low level human players), while the expert human player emphasizes setting up and executing as many multi-line clears as possible.

Because of this and other limitations, the models in their original forms are not very informative about human behavior. However, previous modeling work has found that a basic limitation, imposing a hard limit on the length of a model game, could be employed to induce more human-like strategies in the models (Sibert et al.,2017). A follow-up study showed that the human-like strategy of multi-line clears will arise naturally in models in response to a short game condition, even without explicit reinforcement of the score-seeking behavior (Sibert & Gray,2018).

These results provide a strong argument for the importance of time pressure in shaping human behavior, but so far, it has only been implemented in a very simple way. In this paper, we gift our models with human-like time pressure so that rather than StarTrek-like "beaming" each zoid to its final location in a single instant, moves requiring more zoid movements cost more to execute than those which favor fewer.

Methods

As part of an ongoing exploration of human expertise in Tetris, we have collected gameplay data from over 600 subjects through a combination of laboratory studies where Tetris is played in acoustically isolated "research pods" to local

and international tournaments where players compete against each other in loud and, at times, raucous events. All data was collected using the Meta-T software (Lindstedt & Gray,2015) and its successors, providing access to a huge array of game information including all board states and key presses.

As shown in Figure 2 and Table 2, players were placed into skill bins ranging from Extreme Novice to Extreme Expert based on the mean performance across their best several games. In addition to score differences between groups, players showed a clear shift in strategy from a nearly complete reliance on single line clears (by novice players) to prioritizing multi-line clears (by expert players).

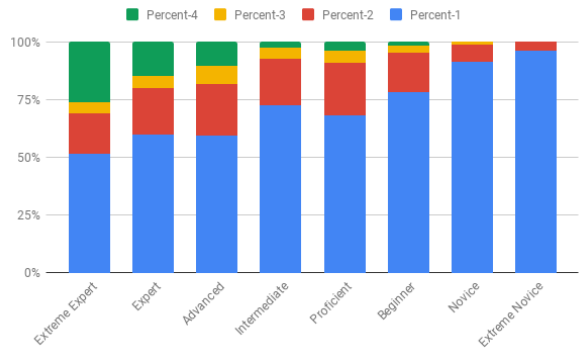


Figure 2: Behavior patterns of human players in each of the eight defined skill bins. The proportion of each bar corresponds to the proportion of each line clear type that players in each group made during their games. Players at the low end of the skill spectrum rely almost entirely on single line clears (blue), while the rate of 4-line clears (green) increases with skill.

Time Parameters

From the raw behavioral data of the human players, we created three measures that capture the execution cost of an individual move: Initial Latency, Average Latency, and Efficiency. These values were logged for all players at every decision point for all games during the experimental period. However, not all these values reflect the actual speed of players. For example, a very advanced player playing at a very slow level may start moving the zoid before a decision is made, and therefore overlapping the costs represented by the Initial Latency and Average Latency parameters (described in more detail below). To account for this, time parameters were not derived from all of a players decisions, but only those made at the Maximum Playable Level, the final level completed before the game was lost. We believe that these decisions are being made at the edge of the player's skill, but are not the random panicked moves often made during the final moments of a game. The parameters for each skill bin are listed in Table 2.

Initial Latency This parameter captures the average time needed for a player to make any key press following the appearance of a new zoid at the top of the screen. When playing to the edge of a player's ability, this initial latency likely captures the processing time to recognize the zoid, some kind of evaluation of possible placement options, the selection of the final placement, and the planning of the keypress sequence required to move the zoid into place. Initial latency is calculated by taking the average time between the zoid's appearance and the first key press for each player. The expertise group value is calculated by taking the average value for all players in the group.

Average Latency This parameter captures the speed of all key presses after the initial key press. As these key presses are likely made after a motor plan has been determined, average latency reflects the motor speed of the player without any decision making or planning. Average latency is calculated by taking the average time between key presses for all but the first key press for each player. The value for each of the 8 expertise groups is calculated by averaging the value across all players in that group

Efficiency A perfectly efficient player will make the minimum number of key presses to move the zoid into position, but humans are prone to all manner of small mistakes that require extra key presses. Zoids are rotated or translated too far and must be brought back into position, players switch paths to pursue an alternate placement, or players simply are unaware that a slightly different order of translations and rotations would require fewer key presses. The efficiency parameter is calculated by determining the optimal path to the final placement and then finding the difference in key presses between this optimal path and the one executed by the player. A player making only the required key presses when maneuvering pieces would have an efficiency of 0. The efficiency parameter is the average number of extra key presses made by a player. The expertise group value is calculated by taking the average value for all players in the group.

Implementing Time Pressure in Models

To capture the effect of time pressure on how long movements could take, we calculated the mean movement time for each type of movement at each of our 8 levels of player expertise (Table 2). In the no time pressure condition, the best placement is calculated and the move is instantly made (This is how all previous Machine Learning Models have worked). The time parameters add the additional step of determining if the move chosen can be made in the time allocated The time cost of a move is calculated by the following formula.

$$TimeCost = InitLat + AvgLat(Path * Eff) \quad (1)$$

The estimated path length (Path) is multiplied by the efficiency parameter (Eff). This extended path length is multiplied by the average latency parameter (AvgLat). Finally, the

Skill Bin	Initial Latency (ms)	Average Latency (ms)	Efficiency
Extreme Expert	79.28	162.11	0.50
Expert	70.56	173.79	0.57
Advanced	96.12	221.72	0.98
Proficient	272.26	433.31	1.64
Intermediate	151.24	311.03	1.16
Beginner	411.29	536.51	1.88
Novice	448.76	668.98	2.06
Extreme Novice	571.37	804.34	2.62

Table 2: Time parameters for each skill bin. Initial and Average latency are measured in milliseconds, and Efficiency is measured in key presses. The Criterion Score is a score based metric reflecting the average of the highest scoring games achieved by players in each group.

initial latency parameter (InitLat) is added to determine the overall time cost (TimeCost).

This time cost is then compared with the zoid drop speed at the current game level. If the move time cost is greater than the time available, the model must choose an alternate move (that does not raise the pile past the top of the screen), and if there are no alternates are possible, the game ends.

Time parameters were implemented into the models in two ways: first, by adding each level of constraints to a high performing model previously trained with no time constraints, and second, by training a model with each set of human-derived time parameters.

Models were trained using the CERL method, reinforced for high score. The model with no time parameters was limited to 525 piece games (a reasonable human game length), but the other models were not explicitly limited in length and relied on the time parameters to restrict game length. The training process ended when the variation of feature weight values dropped below 0.01, and were considered to have converged.

After training, the models were tested by playing a set of ten games using a pre-selected set of game seeds (that would produce the same sequence of zoids). The models were evaluated on their performance, reflected in the scores of these test games, and their behavior, measured by the proportion of line clear types made by the models.

Results

Imposed Time Parameters

Imposing time constraints on a previously unconstrained model caused a significant drop in performance. This performance drop increased as the timecost (see equation 1) became greater. Models slowed to the speed of Proficient players scored hardly any points at all. (See Table 3) The behavior of the model as measured by line clear types remained largely consistent (Table 4), with an emphasis on 4 line clears, until the model was unable to execute line clears of any kind.

Skill Level	Game Length	Lines Cleared	Score
No Time Parameters	409.9 (155.00)	153.1 (67.20)	203766 (133487)
Extreme Expert	278.9 (67.45)	94.4 (26.93)	71878 (32742.14)
Expert	271.1 (64.84)	91.5 (25.92)	68416 (30855.40)
Advanced	231.2 (37.77)	75.9 (15.08)	45000 (12673.27)
Intermediate	116 (51.51)	49.5 (20.14)	23688 (16916.47)
Proficient	41.5 (7.29)	1.6 (1.96)	72 (93.90)
Beginner	24.4 (4.30))	0 (0)	0 (0)
Novice	24.8 (6.03)	0.4 (0.97)	16 (38.64)
Extreme Novice	25.1 (4.95)	0.6 (0.97)	24 (38.64)

Table 3: Performance of models with *Imposed Time Parameters*. Performance is measured by average game length (number of episodes), average lines cleared, and average score. Standard deviations are listed in parentheses.

Trained Time Parameters

Models trained with time constraints were able to score points (see Table 5), even when playing very slowly, though faster models were unsurprisingly higher scoring than slower models. These scores were also roughly equivalent to the average score of human players in the group from which the time parameters were derived. The model behavior also shows a strategy change, with slower models relying almost completely on single line clears, while faster models started shifting toward a multi-line clear strategy. (See Table 4). However, the rate of multi line clears for these models was lower than those of models trained without time parameters.

Discussion

Imposing time pressure on an already trained model resulted in a significant score drop. Indeed, the model was unable to perform at all once slowed to the speed of our mid range players. Unlike humans and unlike models trained at different speeds (Table 6), models trained with no time pressure (Table 4 did not change strategies with changes in drop speed.

In contrast, training models with time parameters produced successful models at all speed levels, and roughly reflected the scores and strategies of the human players in the corresponding expertise groups, with a few notable exceptions. The Extreme Expert group had lower performance than the Expert group because the former has a higher initial latency parameter, making them, on paper, slower. We believe this slowdown is reflective of a higher decision quality by extreme expert players, but as the models all have the same de-

Skill Level	1 Line	2 Line	3 Line	4 Line
No Parameters	20.51% (3.44)	25.48% (7.51)	20.00% (11.36)	34.01% (7.86)
Extreme Expert	20.55% (5.37)	26.63% (9.23)	21.62% (22.34)	31.20% (6.33)
Expert	21.25% (6.03)	26.15% (8.32)	20.51% (11.60)	32.09% (6.27)
Advanced	21.84% (6.47)	27.66% (10.48)	19.52% (10.65)	31.00% (8.65)
Intermediate	22.60% (7.68)	21.21% (12.60)	25.17% (10.66)	31.02% (16.43)
Proficient	73.33% (41.31)	26.67% (41.31)	0% (0)	0% (0)
Beginner	0% (0)	0% (0)	0% (0)	0% (0)
Novice	100% (0)	0% (0)	0% (0)	0% (0)
Extreme Novice	100% (0)	0% (0)	0% (0)	0% (0)

Table 4: Behavior of models with *Imposed Time Parameters*. Behavior is measured by the average percentage of total lines that are cleared with each line clear type (1, 2, 3, or 4). Standard deviations are shown in parentheses.

Skill Level	Game Length	Lines Cleared	Score
Extreme Expert	378 (110.42)	134.3 (44.34)	80350 (45401.84)
Expert	392.1 (90.54)	139.5 (36.03)	95552 (46846.02)
Advanced	337.3 (31.26)	118 (12.45)	32434 (7209.18)
Intermediate	170.1 (64.80)	50.7 (26.05)	23196 (19153.34)
Proficient	115 (80.24)	30.7 (31.50)	4582 (5652.35)
Beginner	134.4 (34.89)	38.6 (13.75)	4514 (3031.51)
Novice	94.2 (34.58)	23.6 (13.66)	2134 (1869.31)
Extreme Novice	66.4 (18.77)	14.1 (7.82)	928 (606.06)

Table 5: Performance of models with *Trained Time Parameters*. Performance is measured by average game L=length (number of episodes), average lines cleared, and average score. Standard deviations are listed in parentheses. For all test games, the game length was unlimited but dependent on the model's speed.

cision ability, the small speed difference results in a slightly lower score. In addition, players in the Extreme Novice group scored much lower than their corresponding model, likely because even when very slow, the models are making better quality decisions than the players. These deviations at the extreme ends of the skill range suggest that a player's skill level is determined by a combination of their speed and their decision quality, rather than just by speed alone.

There were two more deviations between the performance and behavior of the human and the model. First, the Proficient model scored much lower than the corresponding human players. Second, although the Advanced model achieved a score that was close to its human counterparts, it relied primarily on clearing single rather than multiple lines (see Table 4).

The above two cases demonstrate the pitfalls of our model search method. While hundreds of potential models are tested and evaluated during training, the development process is run only once. Under most circumstances, we consider the wide breadth and length of the search sufficient to prevent overfitting, the CERL method can sometimes converge on a locally optimal area of the feature space. Additional noise introduced into the search, or perhaps another search method altogether could possibly mitigate some of these problems and produce more robust models.

Our biggest surprise was the behavior of the time trained models. Though models with faster time parameters displayed higher percentages of 4-line clears, these percentages were lower than the human players in the skill groups from which those parameters were derived, and also lower than the model trained without time parameters. We believe this discrepancy is caused by the implementation of time pressure as a global factor, that is, the models have the same time constraints for a full game and these parameters dictate what is considered a 'good' move based on the extent of the model's ability. This leads the consideration of a 'good' move to be moves that are successful at the final level of the game, where line clears of any type are worth more points. Even highly skilled players, who primarily pursue the score based strategy, shift to the lines based strategy in a bid for survival (Sibert & Gray,2018) at the end of a game. With global time parameters, a move made at level 1 would be given the same score as a move made at level 15, even though at level 15 the time constraints might make that move impossible. Gifting the models with an awareness of the current level might moderate the other features and produce time based models more representative of human behavior.

Conclusions

The deliberate practice framework (Ericsson, Krampe, & Tesch-Römer,1993) identifies practice as the single best predictor of a player's eventual level of expertise. Though the exact definition of *deliberate practice* has proven difficult to pin down, it is most often agreed to be effortful execution of the target task at the very edge of the player's skill. In this way, the current implementation of the model time constraints as global parameters may be forcing the model training into a kind of deliberate practice, and may be forcing our models to use strategies at the highest speed levels that humans would

Skill Level	1 Line	2 Line	3 Line	4 Line
Extreme Expert	34.42% (6.42)	34.48% (5.68)	21.64% (5.34)	9.46% (8.77)
Expert	24.38% (4.13)	40.75% (6.26)	24.42% (5.49)	10.45% (5.10)
Advanced	88.49% (3.66)	10.33% (2.75)	1.18% (1.63)	0% (0)
Intermediate	22.79% (6.49)	34.33% (16.01)	19.24% (9.79)	23.64% (14.56)
Proficient	91.36% (7.17)	8.64% (7.17)	0% (0)	0% (0)
Beginner	88.33% (7.18)	11.19% (7.15)	0.48% (1.51)	0% (0)
Novice	93.28% (8.81)	6.06% (8.31)	0.67% (2.11)	0% (0)
Extreme Novice	90.88% (10.94)	7.45% (8.85)	1.67% (5.27)	0% (0)

Table 6: Behavior of models with *Trained Time Parameters*. Behavior is measured by the average percentage of total lines that are cleared with each line clear type (1, 2, 3, or 4). Standard deviations are shown in parentheses. For all test games, the game length was unlimited but dependent on the model's speed.

not. That is, the (Sibert & Gray,2018) observations show that, while experts mostly favor a multi-line strategy, when playing at the limits of their expertise, humans revert to strategies that favor single-line clears. Our models, in contrast, are more dogged. Once they acquire a strategy they keep with it to the very end. By only practicing, or training (in the model case), under high time constraints, a player may fail to learn or master the alternate strategies that are more successful at earlier levels.

The models provide only a rough approximation of human behavior, but the observed contrasts between them and human players helps to shed light on the relationship between the time-based game constraints and the strategies employed by players. A safe, survival based strategy is best when the game is hard (because of low player skill or high speeds), but true expert players are able to flexibly switch strategies to best suit the current game state. A single, unchanging strategy may be functional in complex, dynamic task environments, but is unlikely to allow for the highest levels of performance.

Acknowledgments

The work was supported, in part, by grant N000141712943 to Wayne D. Gray from the Office of Naval Research, Dr. Ray Perez, Project Officer. The time parameters were originally codified by Mitchell Mellone as a senior thesis project.

References

Belchior, P., Marsiske, M., Sisco, S. M., Yam, A., Bavelier, D., Ball, K., et al. (2013). Video game training to improve selective visual attention in older adults. *Computers in Human Behavior*, *29*(4), 1318 - 1324.

Destefano, M., Lindstedt, J. K., & Gray, W. D. (2011). Use of complementary actions decreases with expertise. In L. Carlson, C. Hölscher, & T. Shipley (Eds.), *Proceedings of the 33rd Annual Conference of the Cognitive Science Society* (p. 2709-2014). Austin, TX: Cognitive Science Society.

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological review*, *100*(3), 363–406. Available from http://doi.apa.org/psycinfo/1993-40718-001

Fahey, C. P. (2015, 09). *Tetris AI.* Available from http://www.colinfahey.com/tetris/ ([Online; accessed 2015-Jan-30])

Gray, W. D. (2017). Games-XP: Action Games as Cognitive Science Paradigm. *Topics in Cognitive Science*, *9*(2), 289–307.

Holmes, E. A., James, E. L., Coode-Bate, T., & Deeprose, C. (2009, 01). Can playing the computer game "Tetris" reduce the build-up of flashbacks for trauma? A proposal from cognitive science. *PLoS ONE*, *4*(1), e4153. Available from http://dx.doi.org/10.1371%2Fjournal.pone.0004153

Kirsh, D., & Maglio, P. (1994). On Distinguishing Epistemic from Pragmatic Action. *Cognitive Science*, *18*(4), 513–549. Available from http://doi.wiley.com/10.1207/s15516709cog1804{_}1

Lindstedt, J. K., & Gray, W. D. (2015). Meta-T: Tetris as an experimental paradigm for cognitive skills research. *Behavior Research Methods*, *47*(4), 945-965. Available from http://link.springer.com/10.3758/s13428-014-0547-y

Lindstedt, J. K., & Gray, W. D. (2019). Distinguishing experts from novices by the mind's hand and mind's eye. *Cognitive Psychology*, *109*, 1 - 25. Available from http://www.sciencedirect.com/science/article/pii/S0010028518300756

Nash, P., & Shaffer, D. W. (2011). Mentor modeling: the internalization of modeled professional thinking in an epistemic game. *Journal of Computer Assisted Learning*, *27*(2), 173-189.

Proctor, M. D., Bauer, M., & Lucario, T. (2007). Helicopter flight training through serious aviation gaming. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, *4*(3), 277-294. Available from http://dms.sagepub.com/content/4/3/277.abstract

Rapp, A., Cena, F., Gena, C., Marcengo, A., & Console, L. (2016). Using game mechanics for field evaluation of prototype social applications: a novel methodology. *Behaviour & Information Technology*, *35*(3), 184-195. Available from http://dx.doi.org/10.1080/0144929X.2015.1046931

Robertson, J. (2003, Nov). A brief history of Tetris AI. Available from http://liquidnarrative.csc.ncsu.edu/index.php/the-news/the-ln-blog/145-a-brief-history-of-tetris-ai?tmpl=component&print=1&layout=default&page=

Sibert, C., & Gray, W. D. (2018). The Tortoise and the Hare: Understanding the influence of sequence length and variability on decision making in skilled performance. *Computational Brain & Behavior*, *1*(3-4), 215–227.

Sibert, C., Gray, W. D., & Lindstedt, J. K. (2017). Interrogating feature learning models to discover insights into the development of human expertise in a real-time, dynamic decision-making task. *Topics in Cognitive Science*, *9*, 1-21. Available from http://dx.doi.org/10.1111/tops.12225

Szita, I., & Lorincz, A. (2006). Learning Tetris using the noisy cross-entropy method. *Neural Computation*, *18*(12), 2936-2941.

Thiery, C., & Scherrer, B. (2009a). Building controllers for Tetris. *ICGA Journal*, *32*, 3–11. Available from http://hal.archives-ouvertes.fr/inria-00418954/

Thiery, C., & Scherrer, B. (2009b). Improvements on learning Tetris with cross-entropy. *ICGA Journal*, *32*(1), 23-33.

Modeling the Effects of Post-Traumatic Stress on Hippocampal Volume

Briana M. Smith (brianam2@uw.edu)
Department of Psychology, University of Washington
Seattle, WA 98195 USA

Madison Chiu (mchiu44@uw.edu)
Department of Psychology, University of Washington
Seattle, WA 98195 USA

Yuxue Cher Yang (chery@uw.edu)
Department of Psychology, University of Washington
Seattle, WA 98195 USA

Catherine Sibert (sibert@uw.edu)
Department of Psychology, University of Washington
Seattle, WA 98195 USA

Andrea Stocco (stocco@uw.edu)
Department of Psychology, University of Washington
Seattle, WA 98195 USA

Abstract

Post-Traumatic Stress Disorder (PTSD) is a psychiatric disorder often characterized by the unwanted re-experiencing of a traumatic event through nightmares, flashbacks, and/or intrusive memories. This paper presents a neurocomputational model using the ACT-R cognitive architecture that simulates intrusive memory retrieval following a potentially traumatic event (PTE) and derives predictions about hippocampus volume observed in PTSD. Memory intrusions were captured in the ACT-R Bayesian framework by weighting the posterior probability with an emotional intensity term *I* to capture the degree to which an event was perceived as dangerous or traumatic. It is hypothesized that (1) Increasing the intensity *I* of a PTE will increase the odds of memory intrusions; and (2) Increased intrusions will result in a concurrent decrease in hippocampal size. A series of simulations were run and it was found that *I* had a significant effect on the probability of experiencing traumatic memory intrusions following a PTE. The model also found that *I* was a significant predictor of hippocampal volume reduction, where the mean and range of simulated volume loss match results of existing meta-analysis. The authors believe that this is the first model to both describe traumatic memory retrieval and provide a mechanistic account of changes in hippocampal volume, capturing one plausible link between PTSD and hippocampus size.

Keywords: Post-Traumatic Stress Disorder; Hippocampus; Amygdala; Declarative Memory; Long-Term Memory; ACT-R; Cognitive Architecture

Introduction

Post-Traumatic Stress Disorder (PTSD) is a psychiatric disorder that originates after experiencing or witnessing a traumatic event, such as rape, domestic violence, assault, a serious accident, or military combat. At the behavioral level, PTSD is characterized by persistent avoidance, alterations in mood, as well as cognitive distortions surrounding the trauma. One of the most characteristic and disruptive behavioral effects of PTSD, however, is the unwanted reexperiencing of the trauma through nightmares, flashbacks, and/or intrusive memories. Traumatic experiences evoke an emotional response that is accompanied by increased activation of subcortical areas such as the amygdala. Intrusive memories are thought to occur because of the simultaneous activation of the amygdala and hippocampus during memory encoding (Marks, Franklin, & Zoellner, 2018). At the subcortical level, PTSD is also characterized by a marked reduction in the volume of the hippocampus—a medial temporal lobe structure necessary for memory functioning. It is important to note that this change is primarily structural, and, although often remarkably apparent, decreased hippocampus size is not accompanied by a functional impairment in long-term memory performance (Karl et al., 2006).

The goal of this paper is to derive predictions about the changes in hippocampus volume observed in PTSD by using a neurocomputational model to simulate intrusive memories over time within an integrated cognitive architecture. The central idea of the model is that intrusive memories operate *within* the context of a general theory of declarative memory, specifically episodic memory. Within this framework, the persistent memory intrusions observed in PTSD can be seen not as a maladaptive response, but rather as the runaway process of an otherwise adaptive memory system.

As a memory is retrieved more frequently, its priority increases and its rate of decay decreases. A traumatic memory, however, tends to out-compete more contextually

appropriate memories due to the fact it was encoded in a highly emotional state. With each retrieval of the traumatic memory, disproportionately more resources are allocated to it, leading to the further preservation and growth of these unwanted memory intrusions. In this framework, it is proposed that the corresponding changes in hippocampal volume associated with PTSD can be explained as the natural result of a biological process to efficiently allocate resources to changing memory demands.

The model presented herein is framed within ACT-R's theory of declarative memory (Anderson, 2007). This choice was motivated by three reasons. First, ACT-R is the most commonly adopted cognitive architecture in psychology and the cognitive neurosciences (Kotseruba & Tsotsos, 2018). Second, ACT-R has a long and established history of application to brain sciences, making the process of drawing new inferences at the neural level easier and less tentative. Finally, ACT-R is based on a Bayesian framework, which provides an elegant foundation of declarative memory retrieval processes and can be easily extended to incorporate the proposed theory of memory retrieval according to their emotional intensity.

The Model

Before introducing the model from a neural and an algorithmic point of view, it is important to frame it within Anderson's analysis of human episodic memory in terms of "Rational Analysis" (Anderson, 1990), or, as it is called currently, Bayesian terms. Throughout this paper, this analysis will be referred to as a guiding principle to modify ACT-R and make inferences about its neural substrates.

In the Bayesian framework, a memory m's probability of being recalled in the presence of a context $Q = \{q_1, q_2, \ldots q_n\}$ reflects the memory's retrieval *need*, and is a Bayesian function of both the past history of m and the degree to which each contextual cue q predicts m. In Anderson's (1990) formulation, the retrieval need of a memory m in a context Q is expressed in terms of a memory's activation $A(m)$, a quantity that reflects its log posterior odds of being retrieved in the presence of Q. Following Bayes rule, the posterior odds can be separated into two different quantities, the prior odds and the likelihood odds:

$$\begin{aligned} A(m) &= \log [P(m|Q) / P(m/\neg Q)] \\ &= \log [P(m) / P(\neg m)] + \log [P(Q|m) / P(Q|\neg m)] \\ &= \log [P(m) / P(\neg m)] + \log \textstyle\prod_q [P(q|m) / P(q)] \\ &= \log [P(m) / P(\neg m)] + \textstyle\sum_q \log [P(q|m) / P(q)] \end{aligned} \quad (1)$$

In ACT-R, it is customary to give different names to the two quantities that make up the right-hand side of Eq. 1, referring to the prior odds as the base-level activation or $B(m)$, and to the likelihood odds as the spreading activation or $S(m)$.

A memory's base-level activation $B(m)$ increases with the frequency of its usage and decreases over time, reflecting the effects of frequency and recency. In ACT-R, each use of a memory m leaves a trace i, and each trace i decays exponentially over time with a decay rate d, which represents an individual-specific rate of forgetting (Sense et al. 2016). A single memory m is associated with multiple traces, each of which corresponds to a time during which m has been encoded, and re-encoded, or retrieved. Thus, the log odds of retrieving m correspond to the sum of the log odds of retrieving each of its individual decaying traces, and $B(m)$ can be expressed as:

$$B(m) = \textstyle\sum_i (t - t_i)^{-d} \quad (2)$$

Spreading activation $S(m)$, instead, can be interpreted in reference to semantic networks, in which memories are connected by associative links, and activation flows through the links to associated nodes in the network. In this case, the activated nodes represent the elements q in the context Q, and the links represent the degree of association or similarity between q and each memory's features. By means of spreading activation, the proper context can facilitate the retrieval of memories whose base-level activation would, otherwise, be too weak. The amount of spreading activation is proportional to the product between the strength of the link connecting q to m (indicated as $s_{q \to m}$) and an *attentional weight*. The weight is usually simplified as a single scalar quantity, W, divided over the number of active elements in the context, N:

$$S(m) = \textstyle\sum_q (W/N)\, s_{q \to m} \quad (3)$$

Different values of W in Eq. 3 alter the degree to which memory retrieval depends on spreading (and, therefore, contextual cues) vs. base-level activation (and, therefore, statistical priors).

ACT-R In the Context of Memory Consolidation

Although ACT-R has been described in many ways, it is useful, given the goal of this paper, to compare it to a prominent neural theory of memory consolidation, the Multiple Trace Theory (MTT: Moscovitch et al., 2005). The MTT assumes that episodic memories originate from distributed representations that span multiple cortical areas (Figure 1). During the encoding phase (red lines in Fig. 1), the different features of an event ($q_1 \ldots q_N$) are encoded by different cortical areas and bound together into a single association map in the hippocampus (as attributes $a_1 \ldots a_N$), through the multiple descending pathways that converge from the cortex through the dentate gyrus. MTT posits that the hippocampus is the permanent store of episodic memories, and that each encoding episode leaves a permanent trace. During retrieval, the hippocampus trace is temporarily re-activated (blue lines in Figure 1) and, through ascending pathways from the temporal lobe to the cortex, causes the re-activation of the original neurons. This reactivation, in turn, might be re-encoded as a second trace.

Base-level activation and spreading activation reflect, therefore, two distinct neural processes. Specifically,

base-level activation reflects processes that are internal to the hippocampal network, such as decay or interference due to accumulation of memory traces (Alvarez & Squire, 1994), while spreading activation reflects the mechanism by which cortical inputs might trigger contextual memory retrieval (Rolls & Treves, 1998).

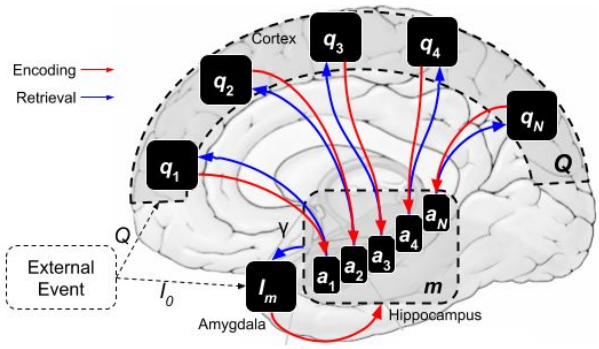


Figure 1: A neuroanatomical interpretation of the model presented herein.

Extending ACT-R to Include Trauma

It has been noted several times, even by Anderson himself (Anderson, 2007, Chapter 3), that one limitation of this approach is that it considers all memories as equally important. On the contrary, not all memories are. Memories of emotional events are thought to persist longer and be more readily available for retrieval than non-emotional memories because of the activation of the amygdala during memory encoding (Marks, Franklin, & Zoellner, 2018). Specifically, memories of events that incorporate threat or fear are of greater importance evolutionarily because they are often critical for survival (Ledoux, 1998). Although some authors have generalized this approach to all emotions (and advanced strong arguments), this paper will limit itself to the responses of the amygdala which are directly connected to PTSD and well understood in neurophysiological terms (Bryant, et al. 2008).

In a Bayesian framework, the concept of survival importance can be easily captured by weighting the posterior probability by an emotional impact term, referred to as *intensity*, $0 < I(m) < \infty$, which captures the degree to which an event was potentially dangerous or traumatic. The posterior odds now become:

$$\begin{aligned} A(m) &= \log [P(m|Q) / P(m/\neg Q)] [I(m)/\mathrm{I}(\neg m)] \\ &= B(m) + S(m) + \log I(m) - \log I(\neg m) \\ &= B(m) + S(m) + \log I(m) - k \end{aligned} \quad (4)$$

The last passage was motivated by the consideration that, over a lifetime, $I(\neg m)$ would approach the mean traumatic value of all memories and thus could be considered a background constant k.

In summary, the proposed Bayesian framework suggests that traumatic events add a *constant bias* that makes a memory more likely to be retrieved, even in the absence of contextual cues and in proportion to the perceived intensity of the traumatic event. In biological terms, this perceived intensity bias can be interpreted as the contribution of the amygdala to hippocampal activation (Fig. 1). The amygdala is bidirectionally connected to the hippocampus and is known to play a key role in processing event salience (Anderson & Phelps, 2001), fear (LeDoux, 1998) and in boosting memory for stressful events (references). Importantly, and consistently with our interpretation, the amygdala is hyper-responsive in individuals suffering from PTSD (Shin, Rauch, & Pitman, 2006).

Deriving ACT-R Predictions For Hippocampus Size

The final step to test this theory consists of deriving predictions about hippocampus size from the augmented ACT-R framework. To calculate hippocampus size, the following analysis was adhered to. In general, it is known that the size of the hippocampus changes with experience. For instance, in a landmark study (Maguire, Woolett, & Spiers, 2006), cab drivers of London were shown to have larger hippocampus volume than the general population. Additionally, another study showed the volume of the hippocampus co-varies with the years of education (Noble et al., 2012). An accepted explanation for this effect is that the size of the hippocampus reflects the biological investment in storing memories that need to be re-used often (Wollet & Maguire, 2011).

An *efficient* memory storing system would encode cells so memories that need to be accessed more frequently use less resources (in neural terms, less cells or synapses) than memories that need to be accessed less often (Huffman, 1952). In the Bayesian terms described above, memories that are accessed more often have the highest *priors* and, in ACT-R terms, the higher *base-level activations*. Knowing the priors of memory utilization, the size of the hippocampus could then be approximated by a measure of the homogeneity of the distribution of the priors. Here, the long-term memory's *information entropy*, H, was utilized, i.e., the quantity (Shannon, 1948):

$$H = -\sum_m P(m) \log P(m) \quad (5)$$

This quantity captures how much information is represented in declarative memory, once the different probabilities of each memory are taken into account. Consider, for example, the case of two London cab drivers who have memorized the same number of addresses but use them with different probabilities. For one driver, all addresses are equally likely to be retrieved, reflecting the fact that his clients are equally likely to request a ride to all of these locations. For the second driver, on the other hand, one single address is requested all the time, while all the others are seldom, if ever, requested by clients. Information entropy is high for the first driver because it is impossible to

predict which address will be requested by the next client. For the second driver, on the other hand, entropy is low, since one memory is highly predictable and all the other can be ignored. Biologically, the first driver needs to allocate more resources (hippocampal cells) to maintain all of these memories than the second, for whom a small number of cells could be used to encode the single memory that predicts most of the clients' rides in their daily routine.

In ACT-R, a memory's probability of being used, $P(m)$, is reflected in its base-level activation (Eq. 1) and, in this paper's specific model, in its intensity $I(m)$. Note the base-level activation $B(m)$ reflects the memory's prior *odds* rather than true probabilities. To translate them into probabilities, base-level activations were normalized across all memories into long-term memory (LTM):

$$P(m) = [B(m) + I(m)] \,/\, \sum_{i \in LTM} [B(i) + I(i)] \qquad (6)$$

Hypotheses and Predictions

Given the theory outlined above, it is hypothesize that (1) Increasing the emotional intensity I of a potentially traumatic event (PTE) will increase the odds of the event memory being retrieved out of context, predicting intrusive memory occurrence observed clinically in patients with PTSD; and (2) Increased intrusion occurrence will result in a concurrent decrease in hippocampal size, driven by the altered landscape of memory recall priors, and thereby capturing the relationship between trauma and hippocampus size.

Methods

To test the hypothesis driving this experiment, a series of computational simulations were run. The following sections describe the details of the simulations

Memory Representations

The simulations described herein differ significantly from most ACT-R models because they focus on modeling episodic memories over extended durations (~6 months) rather than on specific tasks for very short times. Thus, they adopt a uniform memory representation for all memories instead of different, task-dependent structures. Specifically, all memories are vectors of $N = 8$ features. Each feature is given a randomly selected value, called an attribute, from a pool whose size is determined by a given parameter, A (not relevant for this study and thus not discussed). The attributes for all "normal" events are always selected from the same pool, which captures the common features found in one's daily environment. Attributes of PTEs are selected from a different pool of attributes, representing the unique extraordinary features associated with traumatic circumstances.

Model Behavior

The model performs routine behaviors following a perceive-retrieve-respond loop. The loop initiates when a new event occurs in the external world. The event is perceived by the model, and its features are held in sensory buffers that, together, form the current context Q (Fig. 1). The model responds to the current context by first setting a goal to resolve it. When the goal is set, the model retrieves the memory with the highest total amount of activation, $A(m)$. The retrieval process is influenced by three factors: (1) the base-level activation of the model's memories of previous events, $B(m)$, which determines the memories' *priors*; (2) The spreading activation from the current context $S(m)$, modulated by the model's executive attention W; and (3) The intensity $I(m)$ of previous events. This loop captures a simple decision-by-sampling strategy (Stewart, Chater, & Brown, 2012): Facing a new situation, the model responds by retrieving the most contextually appropriate situation faced in the past, balancing recency, frequency, and contextual cues through spreading activation. Once the memory is retrieved, the goal is resolved and a new memory is formed to encode the current event using the contextual cues $q_1, q_2 \ldots q_N$ as its attributes (Fig. 1).

Daily Event Distribution and Simulation Time Window

To model the accumulation of memories in a plausible manner, new events are presented to the model at a frequency that follows a gamma distribution and a realistic daily schedule. On average, the model is presented with approximately ~20 events per day. Events occur between 8:00 AM and midnight, with a peak probability at around noon. This event distribution was chosen to reflect the normal waking hours of a person, with a greater concentration of events during working hours (8:00AM-4:00PM). Each event's emotional intensity I was randomly selected from a uniform distribution between 0 and 2, so that their mean was equal to 1 (and thus the bias term k in Eq. 4 was equal to zero).

Each simulated run of the model lasted 160 consecutive days, starting 100 days before the occurrence of a traumatic event and extended 60 days after that. On the midnight of day zero, a PTE was generated and presented to the model. The intensity of the PTE was explicitly manipulated throughout the simulations, and given the values of $I_{PTE} = 1$ (control condition), 20, 40, 60. The model's time window extended to another 60 days after the PTE.

Dependent Variables

Two dependent variables are the focus of this study. The first is the probability of experiencing an intrusive memory during the day. This is defined as the probability that the model retrieves a memory of the PTE in response to a situation throughout the day. Note that, because the PTE's attributes are different from those of the daily events, its retrieval is always contextually inappropriate, and thus its recall qualifies as *intrusive*.

The other variable is the hippocampal volume reduction, which is measured as a percent change from a control condition. To get a suitable baseline, the average value of *H* (as a proxy for hippocampus size) over the last 10 days of the simulation (corresponding to days 50-60 after the PTE) was compared to the average value of H for the same period of a model run with an identical combination of parameters except $I = 1$.

Simulations

In addition to the intensity *I* of the traumatic event, a number of other parameters were manipulated parametrically. These parameters were derived from a recent review of the PTSD literature (Marks et al., 2018) and reflect idiographic factors that moderate the behavioral outcomes of traumatic stress. They include the vividness of memory re-experience γ; the vividness of sensory encoding, modeled as the size of attributes pool *A*; individual differences in working memory capacity *W* (see Eq. 3); the tendency to ruminate over the traumatic event *R*; and the potential overlap *C* between features of the traumatic event and attributes of daily situations. Although these parameters will not be discussed in this paper, they are summarized in Table 1 and were left in the analysis as they contribute to representative variability in the simulated results. To obtain stable estimates, the model was run 50 times for each of the 576 combinations of parameter values. estimates. In total, the simulations spanned 4,608,000 simulated days, and 103,330,000 simulated events.

Table 1: Model parameters manipulated in the simulations

Parameter	Meaning	Values
I	Intensity of PTE	1, 20, 40, 60
A	Size of attributes pool	6, 8
γ	Vividness of re-experience	0.80, 0.90, 0.95
W	Working memory	4, 8, 12
C	Similarity between PTE and daily events	0, 0.25, 0.5, 0.75
R	Number of rumination events in a day	0, 20

Results

Given the large number of simulations that were run, it is impossible to fully report the complete set of results. For the purpose of this paper, there are two aspects to concentrate on. First, as expected the model does indeed show worse clinical outcomes in response to more traumatic events. Figure 2, below, shows the daily incidence of traumatic memories. A 3x60 ANOVA, using emotional intensity *I* and the days after PTE as factors, revealed that *I* had a significant effect on the probability of experiencing traumatic memories in the days following a traumatic event [$F(2, 1295345) = 37{,}115.6$, $p < .0001$], with higher values of *I* corresponding to higher intrusion probability. Furthermore, *I* interacted significantly with the day [$F(118, 1295345) = 10.3$, $p < .0001$], resulting in different recovery trajectories (Figure 2).

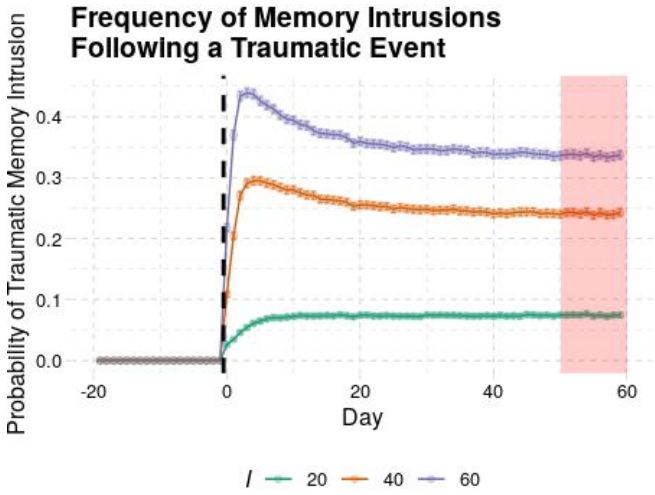


Figure 2: Predicted increase in memory intrusion following a PTE on Day 0 (black dashed line) as a function of emotional intensity *I*. The shaded red area marks the time interval in which the hippocampus volume was calculated.

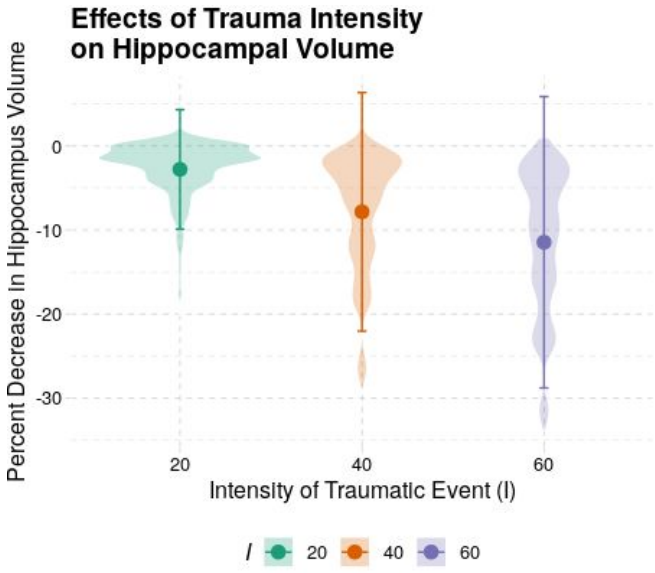


Figure 3: Effect of trauma intensity on hippocampal volume. The violin plots represent the distribution densities of model runs resulting in the corresponding decreases of hippocampal volumes. Solid circles and lines represent means +/- SD.

Having established that the model succeeds in capturing these signatures of PTSD, the results were further examined

to estimate the effects of traumatic stress on hippocampus size. It was observed, across all parameters, that there was general reduction of simulated hippocampus size, ranging from zero to 33.89% with a mean decrease of 7.35% [$t(21,599) = 140.83$, $p < .0001$]. Both mean and range match the results of existing meta-analysis. For example, in Smith's (2005) influential review of structural MRI studies, the range: 0 to 44% and the mean 6.9%. A second question was whether the severity of the reduction was predicted by the severity of trauma. To this end, the model found that the emotional intensity I was a significant predictor of hippocampal volume reduction [$F(2, 21594) = 774.7$, $p < .0001$], with the decrease in hippocampus size growing with greater values of I (all pairwise comparisons significant at $p < .0001$, Bonferroni corrected). This is shown in Figure 3, which shows the distributions of predicted decreases of hippocampal volumes in the simulations, visualized (as violin plots) separately for different values of intensity I.

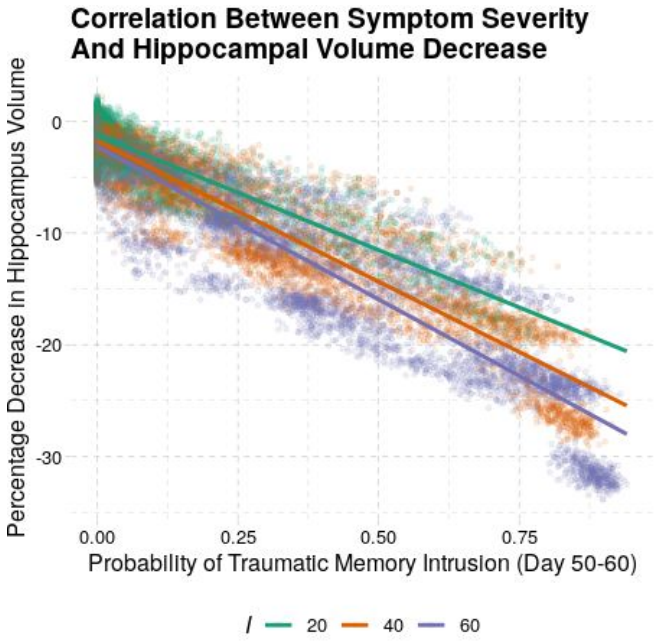


Figure 4: Correlation between the probability of traumatic memory intrusions in hippocampal volume for varying levels of trauma intensity. Each point represents a single run of the model; solid lines represent the mean regression line.

The final analysis investigated was whether or not the degree of hippocampus volume was correlated to the degree of symptom severity. This is important because, although symptom severity is clearly driven by the severity of the traumatic event, it also depends on other factors that were explicitly manipulated in the simulations (see Methods and Table 1). To do so, the mean daily probability of memory intrusions in the last 10 days of the simulations (red shaded area in Figure 2) and the corresponding percentage decrease in hippocampus were calculated for each run of the model. Three separate linear regressions, one for each level of I = 20, 40, and 60, were then computed In all cases, a significant linear regression was found [$I = 20$: $\beta = -20.55$, $t(7198) = -157.9$, $p < .0001$; $I = 40$: $\beta = -25.18$, $t(7198) = -225.6$, $p < .0001$; $I = 60$: $\beta = -27.35$, $t(7198) = 205.1$, $p < .0001$] as shown in Figure 4.

Discussion

This paper has presented a computational model that draws a link between the prevalence of intrusive memories and the changes in hippocampal volume observed in patients with PTSD. To the best of our knowledge, this is the first model to do so, and to provide a connection between Bayesian theories of memories and the underlying neurobiology. The results of this model are also consistent with estimates from the clinical and medical literature. As such, the model may shed light on a number of cognitive factors, such as traumatic memory activation, that contribute to neurophysiological changes associated with PTSD.

In the presentation of this computational model, there are a few obvious limitations. Although an effort was made to account for numerous idiographic factors (see Table 1), it was impossible to account for all various factors that have been deemed clinically important such as age, gender, duration of trauma, recurrence of trauma, comorbidity of other psychiatric disorders, presence and occurrence of other PTSD symptoms, and genetic predisposition. With that in mind, it is feasible that this model can be altered to account for some of these varying factors, as well as other individual differences not aforementioned. Something imperative to take into consideration for future improvements of this model would be, for example, the specific role of the stress hormone cortisol on hippocampal functioning.

These limitations notwithstanding, the model's success in capturing some behavioral and biological factors is encouraging. Theoretically, this model, along with the other research concerning PTSD and its perceived effects on the hippocampus and amygdala, could be used in the future to enhance clinical practice. Targeted, individualized treatments could be developed in which an individual's biological and behavioral measures are used to parametrize a computational model which is, in turn, used to predict long-term recovery trajectories under different medical options.

Acknowledgments

This work was supported by a scholarship from the University of Washington Institute for Neuroengineering (UWIN) to BMS, and partially supported by an award from the Defence Advanced Research Project Agency (DARPA, Grant No. FA8650-18-C-7826). All of the model code, simulation data, and analysis scripts can be found in the Cognition & Cortical Dynamics Laboratory's Github repository at: http://github.com/UWCCDL/PTSD.

References

Alvarez, P., & Squire, L. R. (1994). Memory consolidation and the medial temporal lobe: a simple network model. *PNAS, 91*(15), 7041-7045.

Anderson, A. K., & Phelps, E. A. (2001). Lesions of the human amygdala impair enhanced perception of emotionally salient events. *Nature*, 411(6835), 305-309.

Anderson, J. R. (1990). *The adaptive character of thought*. Psychology Press.

Anderson, J. R. (2007). How can the human mind occur in the physical universe? *Oxford University Press*.

Astur, St Germain, Tolin, Ford, Russell, & Stevens. (2006). Hippocampus function predicts severity of post-traumatic stress disorder. *Cyberpsychology & Behavior:,* 9(2), 234-240.

Bryant, R., Kemp, A., Felmingham, K., Liddell, B., Olivieri, G., Peduto, A., . . . Williams, L. (2008). Enhanced amygdala and medial prefrontal activation during nonconscious processing of fear in posttraumatic stress disorder: An fMRI study. Human Brain Mapping, 29(5), 517-523.

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE, 40*(9), 1098-1101.

Karl, A., Schaefer, M., Malta, L. S., Dörfel, D., Rohleder, N., & Werner, A. (2006). A meta-analysis of structural brain abnormalities in PTSD. *Neuroscience & Biobehavioral Reviews, 30*(7), 1004-1031.

Kotseruba, I., & Tsotsos, J. K. (2018). 40 years of cognitive architectures: Core cognitive abilities and practical applications. *Artificial Intelligence Review*, 1-78.

Maguire, E. A., Woollett, K., & Spiers, H. J. (2006). London taxi drivers and bus drivers: a structural MRI and neuropsychological analysis. *Hippocampus, 16*(12), 1091-1101.

Marks, E. H., Franklin, A. R., & Zoellner, L. A. (2018). Can't get it out of my mind: A systematic review of predictors of intrusive memories of distressing events. *Psychological Bulletin, 144*(6), 584.

Moscovitch, M., Rosenbaum, R. S., Gilboa, A., Addis, D. R., Westmacott, R., Grady, C., ... & Nadel, L. (2005). Functional neuroanatomy of remote episodic, semantic and spatial memory: a unified account based on multiple trace theory. *Journal of anatomy, 207*(1), 35-66.

Noble, K. G., Grieve, S. M., Korgaonkar, M. S., Engelhardt, L. E., Griffith, E. Y., Williams, L. M., & Brickman, A. M. (2012). Hippocampal volume varies with educational attainment across the life-span. Frontiers in human neuroscience, 6, 307.

Rolls, E. T., Treves, A., & Rolls, E. T. (1998). *Neural networks and brain function* (Vol. 572). Oxford: Oxford university press.

Sense, F., Behrens, F., Meijer, R. R., & van Rijn, H. (2016). An individual's rate of forgetting is stable over time but differs across materials. *Topics in cognitive science*, *8*(1), 305-321.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, 27(3), 379-423.

Shin, L. M., Rauch, S. L., & Pitman, R. K. (2006). Amygdala, medial prefrontal cortex, and hippocampal function in PTSD. *Annals of the New York Academy of Sciences*, 1071(1), 67-79.

Smith, M. E. (2005). Bilateral hippocampal volume reduction in adults with post‐traumatic stress disorder: A meta‐analysis of structural MRI studies. *Hippocampus, 15(*6), 798-807.

Stewart, N., Chater, N., & Brown, G. D. (2006). Decision by sampling. *Cognitive Psychology*, 53(1), 1-26.

Woollett, K., & Maguire, E. A. (2011). Acquiring "the Knowledge" of London's layout drives structural brain changes. *Current biology, 21*(24), 2109-2114.

Cognitive Twin: A Personal Assistant Embedded in a Cognitive Architecture

Sterling Somers[1], **Alessandro Oltramari and Christian Lebiere**[1]
[1]Psychology Department, Carnegie Mellon University, Pittsburgh, PA
Bosch Research and Technology Center, Pittsburgh, PA
{sterling@sterlingsomers.com,Alessandro.Oltramari@us.bosch.com, cl@cmu.edu}

Abstract

This paper presents an analysis of a cognitive twin, implemented in a cognitive architecture. The cognitive twin is intended to be a personal assistant that learns to make decisions from your past behavior. In this proof-of-concept case, we have the cognitive twin select attendees to a party, based upon what it has learned (through ratings) about an agent's social network. We evaluate two versions of a model with respect to rate of change in the social network, the noise in the rating data, and the sparsity of the data.

Keywords: cognitive architectures; personal assistants; act-r;

Introduction

Smart recommendation systems, automated alerts, digital personal assistants, and the like have become ubiquitous in our everyday lives. From speakers in our living rooms, watches on our wrists, and, of course, phones in our pockets, we are constantly being notified and updated with information that, at times, can even be helpful. Beyond these current efforts, we envision a cognitive twin: an automated personal assistant that knows the kinds of decisions you would make and uses that knowledge to carry out tasks in the digital world on your behalf. We instantiate our cognitive twin in a cognitive architecture that exhibits human-like cognitive constraints that, for better or worse, can result in biases such as recency and frequency effects. We aim to show that cognitive constraints like these can actually be beneficial in a dynamic environment.

Existing technology requires large data sets which often includes aggregate data from multiple people. There is no doubt that useful inferences can be made by learning from aggregate data, however, there are limits in of the effectiveness of big-data approaches: sometimes the choices you would make are not the same as the average person. Furthermore, these approaches gather data about everyone in a central location which raises concerns about privacy. In our approach, we advocate learning about you from your own data. We cast your behavior as decision that had desirable or undesirable outcomes. That information is stored in the memory of your cognitive twin and is used to make future, similar decisions. Since the decision maker whom we are attempting to model is a cognitive agent, with limited cognitive capacities, we instantiate our model in cognitive architecture that is used in the cognitive sciences to make scientific models of the human mind. The resultant model is human-like and is personalized to make decision for you, negotiate decisions with other cognitive twins, without sharing your data, and without storing your data on a central repository.

In this work we evaluate a prototype cognitive twin developed in the cognitive architecture, ACT-R (Anderson, 2007) using instance-based learning (Gonzalez, Lerch, & Lebiere, 2003). Data is generated from a discrete action simulation of a population of simulated agents who carry out tasks. For each artificial agent in the simulation, a log of their activity is generated. We then use these logs as input to generate a cognitive twin for each simulated agent. In this context, the cognitive twin, therefore, is a simulation of a simulation. As the simulation itself is a prototype, we do not assume a high degree of veridicality. Instead, we use the simulation as a data generator, and evaluate the cognitive twin with respect to that world.

In a machine learning context, it may, at first glance, seem like an misguided choice to use an architecture that is specifically designed to have human-like cognitive constraints. A learning model that, in effect, will 'forget' data over time may seem like a waste of data. However, in a dynamic environment, data can become outdated, as targets change, you might not want to make the same decision you made in the past, even if that decision had a positive outcome. In the present study we test the model's ability to deal with change in two ways. First, directly, we modify the rate of change in the simulation (α) and expect the model to do well with moderate rate of change. Second, we test how many weeks worth of simulated data we use in the model. By reaching back, n-weeks, we hope to show that the learning mechanism gracefully degrades old data; as well as being able to show that the model is robust to few learning examples.

In the spirit of testing real-world-like scenarios, we also test to see how robust the model is to noisy data and sparse data. We test the sensitivity to noise directly by modifying the noise parameter in the simulation (σ). We test sparsity in two ways: first, by using n-previous-weeks worth of data (same as above); and second, by probabilistically controlling how much simulated data is 'seen' by the model (effectively removing data probabilistically).

Background

In previous work (Somers, Oltramari, & Lebiere, 2020) we tested two versions of the cognitive twin, central and dis-

tributed, against one another and against the simulation in their respective success at inviting guests to a 'dinner party'. In this section we describe the dinner-party-planning scenario and the simulation.

Dinner Party Planning

Think about the knowledge that goes into something as seemingly simple as planning a dinner party. You need a place, a time, food, and friends. The time has to suit the attendees (scheduling) and should be during 'dinner' (knowledge), the food should suit the attendees (dietary restrictions), and the people who attend should all get along (unless you are hosting a fight club). Presumably, when you carry out the task of planning a dinner party, you do so partly from knowledge about how to carry-out the task (task-knowledge), partly from things you have learned (who you like, what foods people eat), and partly from communication with other attendees (what food they eat and who they like). Task knowledge can be difficult to learn from data because those processes are often communicated at least in part explicitly as instructions, in a process known as Interactive Task Learning (Laird et al., 2017). We advocate a hybrid approach in which we combine structured knowledge with and data-driven knowledge.

In our previous (and present) work we evaluate our model on its ability to select party guests based on past experiences with them and communicating with other cognitive twins (in distributed model). The simulation assumes a social rating system: when simulated agents interact with one another, they provide a 'rating' of that interaction. That rating is then used to update their social link (their friendship). We describe the simulation, the social interactions, and the rating system more thoroughly below.

Simulation

To generate data, we implemented a discrete simulation. While the simulation, itself, is currently under development, we believe it has enough complexity to generate data to test an early prototype of the cognitive twin. We do not assume that the simulation is veridical on any meaningful dimension but use it, instead, to generate data in lieu of human data. As we progress, we fully intend to develop the simulation further, however, in this analysis we are more concerned about evaluating the cognitive twins, with respect to properties we believe would effect its performance due to underlying cognitive assumptions.

The simulation is comprised of a 'world', populated by agents going about their daily lives. In our previous work our world was comprised of 100 agents, whereas in this analysis, we have doubled the number of agents to 200 (largely to add stability to social networks). Importantly, the agents in the simulation are distinct from their cognitive twins. The agents in the simulation represent 'real' humans (in lieu of real human data) and the twins use the data they generate to plan party attendees. The simulation creates a population of agents by first generating families with sizes controlled by a weighted distribution (parameter). A family is generated, populated, and then new families are generated and populated until the desired population has been reached. Although the analysis we do in this paper is with regards to social interactions, social dynamics between family members are not yet simulated any different than social interactions with other members of the population. We hope to account for differences in social dynamics where appropriate in future work.

Since our proof-of-concept scenario is dinner-party planning, we have developed the simulation to include: social interactions (to develop social links between agents), daily activities (scheduling concerns), and food consumption (to model dietary restrictions). Given space constraints, we will only discuss social interactions (the main focus of this paper).

Social Interactions Each agent in the simulation has both incoming and outgoing social links, that range between -1 (extreme dislike) to +1 (extreme friendliness), to other agents. When a world is generated, the social links are selected from a truncated normal distribution. The mean and standard deviation of the distribution are free parameters but currently set to a mean of 0.4 (under the assumption that social links are generally positive) and a standard deviation of 0.5 (wide enough that negative relationships can still occur). The social links are independent and, as a result, can be asymmetric (i.e. one agent may favor another but may not be favored in return). While negative values represent dislike and positive values represent positive feelings, we explicitly divide positive links into two classes: positive relationships (links greater than zero but less than 0.50) and friends (links greater than or equal to 0.50). In our previous work we set that split at 0.75 but due to changes in the simulation (modified how links change), we are able to better track changes in friendship circles over longer periods of time, by including more people. The initial seeding of social links is used during run-time when setting up social interactions. Roughly, friends become more likely to interact and enemies less likely, as described below. Social interactions (parties) are created during run-time at the start of each simulation day. Each party has a host and each host is selected randomly from the population. The parameter p is set as a proportion of the total population from which the hosts are selected. In our previous work and in the current work we used a value of 0.25 to ensure that we are generating sufficient data to test the model. Modifying this parameter would effect the rate of change in the simulation but we have kept it steady in favor of a parameter that affects the change more directly. Once we have selected hosts for the parties, attendees are selected. Attendees to each party are selected based on their connection to the host. The outgoing links from each center, to each individual in the population is transformed into a weighted probability distribution such that stronger links result in higher likelihood of being selected. The outgoing links from the host are transformed with the following considerations: a) links lower than a minimal value of -0.1 (a free parameter) are excluded; b) a small bias (additive, 0.3, also a free parameter) is added to weights for connections above the 'friend' threshold. These parame-

ters were set qualitatively in previous work to produce steady world statistics.

Even though 25% of the population are chosen as party hosts, not all hosts will result in a social interaction. Once the attendees are selected, the party has to be scheduled and, due to scheduling conflicts, prospective parties may be canceled. For each potential party, a minimum party size is selected randomly from a truncated normal distribution (a free parameter). If the party does not meet that minimum size, the social interaction is canceled. Social interactions are resolved in a queue, which is ordered by the original selection of hosts. Attendees who become committed to a party that is resolved early in the queue may be too busy to attend parties further in the queue. Finally, it is worth noting that although people with large negative connections with the host will not be scheduled by the host, people with negative links could find themselves at someone else's party. The simulation does not try to maximize the overall average connection between guests.

During an interaction, all agents in attendance receive an interaction score with all other guests. In the current work, the interaction score is: the mean links between the agents (incoming and outgoing) plus a randomly selected score value. The score values are selected from a truncated normal distribution between -1 and 1 and the shape of that distribution is a free parameter. We simplify this work from previous work, setting the mean of that distribution to 0.0 (instead of a positive value). In this work, we modify the parameter, σ, which represents the standard deviation of the score value distribution. We consider higher levels of σ as higher levels of noise in the ratings. In the analysis below, we test the robustness of the models to different levels of σ. The final score (mean + score value) is truncated to a range of -1 to 1. This form of scoring is an update from previous work. The social interaction scores are independent: agent A will score the interaction with agent B differently than B will score the interaction with A.

Change in Social Networks The social interactions are the means through which social networks change within the simulation. In this work we have updated how the social links are updated in response to social interactions. We introduce a new parameter, α to represent a rate of change in the simulation that we then systematically modify to test our models against. Social links between agents are updated with the following equation: $L_t = \alpha \cdot L_{t-1} + 1 - \alpha * score_t$, where L_t is the outgoing link at time t and *score* is the interaction score. Higher α values should result in small changes in the networks (because your links are only marginally affected by interaction score), whereas small values of α should result in higher rates of change in social network.

In the present work, we test the robustness of the model at different values of both α and σ. By modifying these two parameters we create low- and high-change simulation conditions (α), and low- and high-noise conditions (σ), creating 4 categories: low-change-low-noise, low-change-high-noise, high-change-low-noise, and high-change-high-noise.

Simulation Analysis

Figure 1 presents different measures of the simulation at each of the four parameters settings: $\alpha = \{0.95, 0.25\} \times \sigma = \{0.25, 0.75\}$.

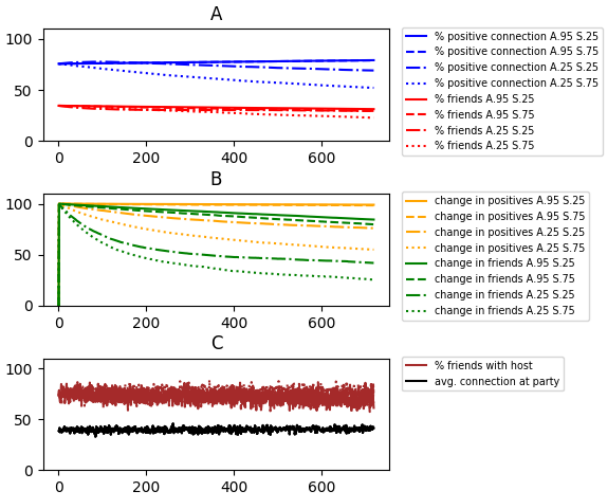


Figure 1: Summary measures of the simulation at different parameters of α and σ. A) shows the world averages of percent positive connections at each parameter settings (blue) and the world average percent of friends (red). B) Illustrates the change in the network from simulation start through two years for both positive connections (orange) and and friendships (green). C) Illustrates the the measures of party success: percent of people who are friends with the host (brown), and the average link between party attendees (black).

Figure 1-A shows the world average percent of positive connections over two years of simulation (blue) and the world average percent of friends, i.e. positive connections greater than 0.50, (red). This plot suggests that, other than $\alpha = 0.25, \sigma = 0.75$n (high-change-high-noise), changes in social networks are not due to a net loss positive connections. The net loss of positive connections (dotted blue line), demarcates a quality boundary, and, therefore provide a good point to limit the adjustment of the α and σ parameters. We expect, in the high-change-high-noise condition, that the cognitive twin would score lower due to overall loss in positive connections.

Figure 1-B illustrates the change in the social networks from time zero through two years of data. Orange lines denote changes in networks of positive connections, while the green lines denote changes in networks of friends (social links greater that 0.50). The change in the networks decay as you might expect: an increase of change with a smaller α and an increased change with increasingly noisy ratings. And, of course, the effect combines, with the greatest rate of change occurring with changes in both α and σ.

Finally, Figure1-C shows the scoring the simulation gets with respect to the percentage of guests who are friends with the host (brown) and the average connection between all guests at a party (black). All parameters are plotted but, due to the fact that those values are somewhat noisy, and the overlapping lines, there is no appreciable difference at different parameter settings. However, they do provide a ball-park estimate how well the simulation does on those measures (the same measure of the model, below). These are the measures we use to evaluate the models.

Cognitive Twin Model

At a high-level, the cognitive twin is an automated decision maker. It uses your data to make the kinds of decisions you would make. In the party-planning scenario, your cognitive twin is intended to do some of the leg-work of party planning for you. The cognitive twin was developed in ACT-UP (Reitter & Lebiere, 2010), a toolkit implementation of ACT-R, developed to make it easier to integrate with simulations, with limited footprint, and network-size scaling requirements. It leverages the key equations for declarative memory, reflecting tradeoffs in recency and frequency in generalizing patters of user activity. Finding the most compatible set of guests relies on ratings of past social activity. The ratings generated by the simulation are represented in the model as chunks in memory consisting of: a) the agent with whom the user interacted, and b) the rating of the interaction (described above). Each of those chunks have an associated base-level activation that reflects the recency (and frequency if the same rating between the same agent has been provided multiple times) of the ratings. Each chunk activation can therefore be interpreted as the relative importance of that rating among competing ones, and factored accordingly by memory retrieval processes. Specifically, the blending retrieval process (Lebiere, 1999) is used to produce a consensus estimate of social rating between two agents by retrieving and aggregating individual ratings weights according to their activation.

These social compatibility ratings are used in a greedy algorithm that starts with the central user organizing the gathering. The rating of all potential guests are evaluated against the current set of invitees, starting with the host, and the guest with the highest rating is selected. In the centered version of the algorithm, the selection process is simply repeated until the guest list is full. In other words, only the social rating between potential guests and the host are taken into consideration. In the distributed version of the algorithm, however, after each guest is selected, the social ratings between that guest and the rest of the potential guests are added in, and the next guest is selected. Again, the process repeats until the guest list is full. Note that it is possible that this process does not yield an optimal outcome because the sequential selection process could lead to a local optimum. For instance, it could select a guest that has the highest social rating with the current guests but very poor social ratings with all remaining potential guests.

Model Analysis

Not only do we test the models with respect to the α and σ parameters in the simulation, we also analyse how robust the models are data sparsity (weeks, probability), and a base-level learning parameter in architecture.

Base-Level The base-level learning parameter controls the rate of power law decay in memory. Memory chunks always remain present in memory, but their activation gradually decays, unless it is boosted again by another experience or rehearsal, reflecting the power law of practice. Since activation controls the retrieval of chunks, activation decay slows and ultimate prevents access to those memories (forgetting) as well as reduces the salience of the chunks in the blending process. While forgetting seems like an unwanted characteristic in an intelligent social agent, it constitutes an adaptive mechanism to the rate of change in the environment, as hypothesized by the rational analysis of cognition (Anderson, 1990; Anderson & Schooler, 1991). As attitudes change and old events becomes less relevant as time passes, favoring more recent information through gradual decay of older memories provides a rational approach to optimizing one's interactions with the world.

We test the effectiveness of the base-level in a number of ways. First, we test how well it responds to modifications of α and σ (which both increase the rate of change, though for different reasons). Second, by modifying the n-previous weeks worth of data, we include increasingly older data in the model. In the high-change conditions, we would expect old data to naturally degrade the twin performance (if it did not have base-level activation) because the social networks have changed, and agents that have interacted long ago may longer be friendly (and visa versa with enemies). Because base-level decay gracefully reduces the impact of older data, we expect the model to be robust, even as we increase the data to the full two-years. Finally, We also test the base-level parameters (BLL) explicitly, specifically In this analysis we start at a generally accepted level of $BLL = 0.5$, and increase it gradually, to explore how well it responds to the increased rate of change.

We also test the robutness of the models with respect to data quantity in two ways described below: n-previous-weeks and probability.

n-Previous Weeks

In our previous work (Somers et al., 2020), we examined how robust the model was to different amounts of data. The simulation at that time, did not have the same social change function (with parameter α). We run a similar analysis here by running the model but including only the last n weeks of data. Figure -(1) and -(2) shows the results of running the model with 1-, 2-, 4-, 8-, 26-, 52-, and 104-past weeks of data.

As expected, the central version of the model (solid) outperforms the distributed model (dotted) in the percent-

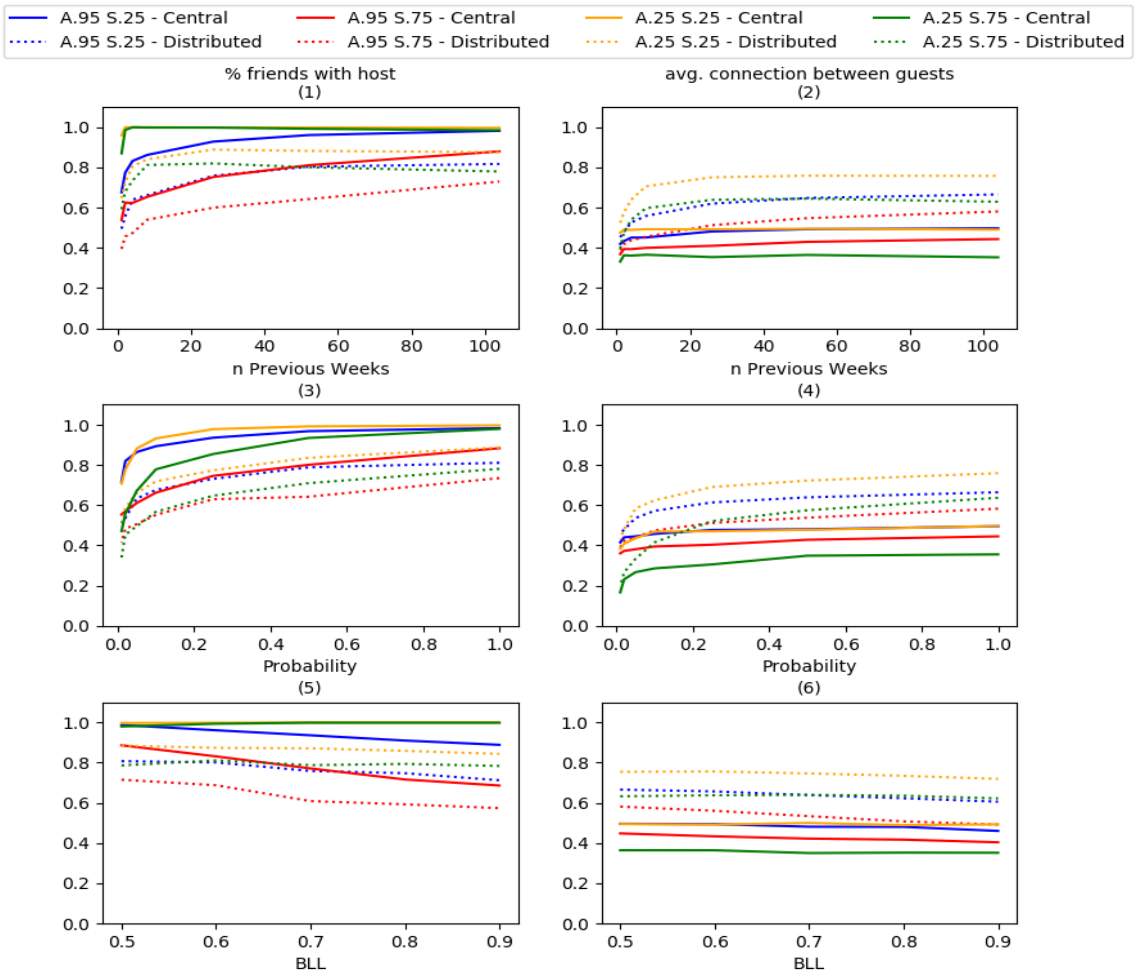


Figure 2: Central (solid) vs. Distributed (dashed) models for each combination of α and σ. The measures are the percent of friends in attendance at a party (left column) and average connection between guests at a party (right column), for *n* previous weeks worth of data (top), probability of including a rating (middle), and the base-level learning parameter of the model (bottom).

friends-with-host (PFH) measure (expected because the central version maximizes for the host, and the distributed maximizes for all attendees). Note, reading left to right on the "n Previous Weeks", although exhibits an increasing amount of data, is actually backwards with respect to time with n-previous being the weeks previous to the end of the simulation. This is important because the simulation is, generally, more stable later in the simulation. In general, the model is quite robust with a small amount of data, with perhaps a slight dip in performance in PFH, in the simulations with the most change (yellow and green), as data reaches back further. In general, the model improves with more data, except in the highest change conditions (yellow and green). In the yellow and green simulation conditions (high change), there is no noticeable improvement in attendees beyond two previous weeks worth of data. Note that although there is no increase, the important trend is limited loss (we want the base-level to naturally degrade old information). In that light, we see a very limited dip in performance at 104 weeks worth of data, suggesting the base-level is mitigating effects of old data in a dynamic environment.

Both the central and distributed model show relatively poor performance in the low-change conditions (blue, red), with few n-previous weeks of data, in the PFH measure. The models exhibit a gradual increase in performance with more data. In these conditions there is very little change to account for, so the base-level appears to over-compensate, benefiting from old data.

In the average-connection-between-guests (AG) measure - (2), the distributed model outperforms the central model (as

expected). The performance of the central model is comparable to the simulation itself (see Figure 1-(c), black lines), with average connection between guests just below the 'friend' threshold (0.50). The central model shows negligible change with increasing number of weeks' data. The distributed models generally improve up to twenty weeks worth of data. There is a fairly strong benefit for the distributed model in the $\alpha = 0.25, \sigma = 0.25$ condition which may indicate a local maximum: a high rate of change (which is taken advantage of by the base-level learning) and a low amount of noise. Beyond 52 weeks, the average connection between friends is above the 'friend' threshold (0.50).

The trend in the AG measure is, overall, similar, showing fairly flat results, suggesting robustness to outdated data in a dynamic environment. There is a slight increase, again, in the low-change conditions (blue, red), for the distributed mode, suggesting, again, that in a static environment, the base-level overcompensates, but recovers with increased data.

Probability

Another measure of data sparsity, one that is probably more naturalistic, is to exclude data probabilistically. We would expect, in reality, to have imperfect participation in our assumed social rating system. In this analysis we encode data in the model probabilistically, simulating agents not providing ratings. Figure -(3) and -(4) illustrates the results of running the model with data probabilities 1.0, 0.5, 0.25, 0.1, 0.05, 0.02, and 0.01. Note that values in this analysis should have the same results at a probability of 1.0 as the weeks analysis had at 104 weeks (they are the same conditions).

Again, central and distributed models preform as expected relative to one another in the respective measures (PFH vs AG). Unlike weeks data, the models typically prefer more data. This difference in amount of data preference is expected because the data is lost evenly with respect to time, so there is no benefit or detriment of the base-level activation. Note that the low-change conditions (blue, red) respond similarly to increased data as in the n-previous weeks. This helps to confirm that the lower performance of the models in those conditions is due to overall data loss, suggesting that those conditions are static, and would benefit from increased data, regardless of how old the data is (as in the n-previous weeks analysis).

The values in both the PFH and AG measures are not too dissimilar from the weeks data, though, notably, the models are not as robust to noise (i.e. red vs blue, green vs orange). The noisy conditions (red, green) perform more poorly in this condition than they had in the n-previous-weeks condition. The lowered performance does make sense, when you consider that in the n-previous-weeks condition, data was removed in blocks, allowing trends to be captured; whereas in the probability condition, data is lost spuriously, amplifying the noise, as might be expected.

Base-Level Learning

In the base-level learning (BLL) analysis, we modify the base-level learning parameter in the architecture from generally accepted range of 0.5, to high value of 0.9 in increments of 0.1. Like the previous analysis, this model plans attendees after the two years of simulation data.

As expected, the central model performs better in the PFH measure (-(5) and -(6)). There is no noticeable change in the PFH scores for the central model, in the highest change conditions (orange, green). The central and distributed model show a decrease performance with an increase in BLL, indicating that increasing the decay rate has an amplified, detrimental effect in a static environment.

In this AG measure, the models are relatively unaffected by the BLL increase. There is not much striking here, other than general inferences, discussed in the next section.

Conclusions

Despite the mildly lower performance in the probability condition, compared to the n-previous-weeks, overall performance of the model is really quite high, especially in comparison to the scheduler in the simulation. In all conditions, other than the high-change-high-noise conditions, the model outperforms the simulation scheduler and in many cases, by a large margin. The distributed model performs especially well scoring reasonably well in both n-previous-weeks and probability conditions, for both PFH and AG measures.

With with respect to change, the model generally prefers a high-change environment (yellow/green vs blue/red) though it responds more poorly with noisy data (e.g. green vs. orange). This is further evidenced in the BLL conditions, with the model scoring more poorly in low-change (blue/red) with increasing BLL parameter. In the low-change conditions, it suggests the cognitive constraints could be a detriment. Looking at Figure 1, however, the low-change conditions are perhaps unrealistically low, showing very little change in social networks over the course of two-years. Future work could explore lowering the BLL parameter in the model to better deal with low-change situations.

Perhaps the most straightforward conclusion from the analysis is that the model is fairly robust to sparse amounts of data. In both n-previous-weeks and data-probability cases, the models perform quite well (in comparison to the simulation). While the models generally show improvement with more data, there is a steep jump (in most cases) with a mild increase in data.

Developing our cognitive twin in a cognitive architecture that is limited by human-like cognitive capacities, though at first glance may seem bizarre, given reliance on big-data approaches, is evidenced to naturally manage data in a dynamic environment. In a dynamic environment, old data eventually may become outdated and detrimental. We have shown that by developing a hybrid learning system, embedded in a cognitive architecture, can be robust to change and data loss.

References

Anderson, J. R. (1990). *The adaptive character of thought*. Psychology Press.

Anderson, J. R. (2007). *How Can The Human Mind Occur In The Physical Universe?* New York, NY: Oxford University Press.

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological science*, *2*(6), 396–408.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*(4), 591–635. doi: 10.1016/S0364-0213(03)00031-4

Laird, J. E., Gluck, K., Anderson, J., Forbus, K. D., Jenkins, O. C., Lebiere, C., ... Kirk, J. R. (2017). Interactive task learning. *IEEE Intelligent Systems*, *32*(4), 6-21. doi: 10.1109/MIS.2017.3121552

Lebiere, C. (1999, 06). The dynamics of cognition: An act-r model of cognitive arithmetic. *Kognitionswissenschaft*, *8*, 5-19. doi: 10.1007/s001970050071

Reitter, D., & Lebiere, C. (2010). Accountable modeling in act-up, a scalable, rapid-prototyping act-r implementation. In *Proceedings of the 10th international conference on cognitive modeling, iccm 2010* (pp. 199–204).

Somers, S., Oltramari, A., & Lebiere, C. (2020). Cognitive twin: A cognitive approach to personalized assistants. In *Aaai spring symposium: Combining machine learning with knowledge engineering (1).*

Connecting Biological Detail with Neural Computation: Application to the Cerebellar Granule-Golgi Microcircuit

Andreas Stöckel (astoecke@uwaterloo.ca)[1]
Terrence C. Stewart (terrence.stewart@nrc-cnrc.gc.ca)[2]
Chris Eliasmith (celiasmith@uwaterloo.ca)[1]
[1]Centre for Theoretical Neuroscience, University of Waterloo
[2]National Research Council of Canada, University of Waterloo Collaboration Centre
200 University Avenue West, Waterloo, ON N2L 3G1 Canada

Abstract

Neurophysiology and neuroanatomy limit the set of possible computations that can be performed in a brain circuit. Although detailed data on individual brain microcircuits is available in the literature, cognitive modellers seldom take these constraints into account. One reason for this is the intrinsic complexity of accounting for mechanisms when describing function. In this paper, we present multiple extensions to the Neural Engineering Framework that simplify the integration of low-level constraints such as Dale's principle and spatially constrained connectivity into high-level, functional models. We apply these techniques to a recent model of temporal representation in the Granule-Golgi microcircuit in the cerebellum, extending it towards higher degrees of biological plausibility. We perform a series of experiments to analyze the impact of these changes on a functional level. The results demonstrate that our chosen functional description can indeed be mapped onto the target microcircuit under biological constraints. Further, we gain insights into why these parameters are as observed by examining the effects of parameter changes. While the circuit discussed here only describes a small section of the brain, we hope that this work inspires similar attempts of bridging low-level biological detail and high-level function. To encourage the adoption of our methods, we published the software developed for building our model as an open-source library.

Keywords: biologically plausible spiking neural networks; Dale's principle; Neural Engineering Framework; delay network; cerebellum; NengoBio

Introduction

Human cognition is ultimately grounded in neurophysiological processes. As suggested by Marr's "levels of analysis" (Marr & Poggio, 1976), cognitive scientists tend to implement models of cognition at algorithmic and computational levels, without explicitly taking limitations of the underlying neural substrate into account (Eliasmith & Kolbeck, 2015).

Depending on the hypothesis that is being explored, ignoring biological detail can be reasonable. Yet, a closer look at biology may help in two complementary ways. First, we can *validate* hypotheses about cognition by determining whether a particular algorithm can be implemented using the constraints of the biological neural network in question. Second, we can *generate* new hypotheses by asking what class of algorithms a particular neural network could support.

We believe that cognitive modelling must ultimately embrace a combination of these two approaches to narrow down the vast space of possible cognitive science theories and to direct research attention within that space. However, a central roadblock to the adaptation of such methods is the availability of modelling tools that make it possible to specify detailed biological constraints (e.g., neural response curves, spike rates, synaptic time constants, connectivity patters) while still being abstract enough to facilitate the specification of high-level cognitive function.

One approach designed to help bridge this gap is the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003), in conjunction with the related Semantic Pointer Architecture (SPA; Eliasmith, 2013). Up until recently however, it has been unclear how to incorporate certain biological constraints that are often described in the neuroscience literature into NEF networks. For example, and despite initial progress in this direction (Parisien, Anderson, & Eliasmith, 2008), accounting for Dale's principle with purely excitatory and inhibitory neuron populations, as well as incorporating spatial connectivity constraints, has been relatively challenging with the existing NEF-based software tool, Nengo (Bekolay et al., 2014). Furthermore, certain aspects of the NEF, such as the neural bias currents J_{bias} have a somewhat unclear relationship to biology.

In this paper, we describe recent advances in modeling techniques that partially alleviate the shortcomings of the NEF listed above. We then apply these methods to extend a previous biologically detailed model of the Granule-Golgi circuit in the cerebellum (Stöckel, Stewart, & Eliasmith, 2020). In this way, we validate that—at least for the set of constraints considered in our experiments—the Granule-Golgi circuit is indeed well-suited to implementing a specific algorithm for encoding temporal information using basis functions. Furthermore, we generate possible hypotheses as to why various biological parameters (such as the sparse connectivity patterns and the time constants of the neurotransmitters) are as observed. We have released the open-source tool we developed to encode these constraints as an add-on to Nengo called NengoBio.[1]

The remainder of this paper is structured as follows. We first review the high-level function we hypothesise could be implemented in the Granule-Golgi circuit, as well as the particular neurophysiological constraints of this network. We then discuss five neural network implementations with an increasing amount of biological detail, along with the corresponding extensions to the NEF. Finally, we perform a series of experiments that explore the impact of individual parameters on the performance of the increasingly realistic system.

[1]See `https://github.com/astoeckel/nengo-bio`.

Background

In order to explore the consequences of adding biological details to a neural system, we need to choose the desired computation that the neural system should ideally perform. In machine learning, the simplest artificial neural networks are purely feed-forward, i.e., they possess no backwards-directed or *recurrent* connections. It is well-known that such neural networks are universal function approximators (Hornik, Stinchcombe, & White, 1989). That is, given enough neurons, any function $f(x) = y$ can be implemented as a neural network by simply having a single hidden layer of neurons that receives x as an input (via a set of input weights) and produces y as an output (via a set of readout weights).

Neurobiological systems differ from the artificial neural networks mentioned above in two key aspects. First, they are intrinsically dynamical systems, i.e., input and output are functions over time. Second, they often include recurrent connections. As has been shown by Eliasmith and Anderson (2003), adding recurrent connections along with their associated synaptic dynamics allows for the creation of neural networks that approximate any differential equation of the form $\frac{d\mathbf{m}}{dt} = f(\mathbf{m}, u, t)$. Again, with a sufficient number of neurons and the corresponding connectivity, such differential equations can be approximated to any desired degree of accuracy. Our goal in this paper is to explore how well such computations can be performed in the presence of other biological constraints.

The Delay Network

As a benchmark for evaluating this performance, we have chosen the following linear differential equation.

$$\begin{aligned} \theta\dot{\mathbf{m}} &= \mathbf{A}\mathbf{m} + \mathbf{B}u, \quad \mathbf{A} \in \mathbb{R}^{q\times q}, \mathbf{B} \in \mathbb{R}^{q}, \mathbf{m} \in \mathbb{R}^{q} \\ (\mathbf{A})_{ij} &= \begin{cases} (2i+1)(-1) & i < j, \\ (2i+1)(-1)^{i-j+1} & i \geq j, \end{cases} \\ (\mathbf{B})_i &= (2i+1)(-1)^i. \end{aligned} \tag{1}$$

This equation is derived by taking the Padé approximate of the continuous-time delay $F(s) = e^{-\theta s}$. As such, this differential equation stores the past history of its inputs the state variable $\mathbf{m}$ (Voelker & Eliasmith, 2018). That is, given $\mathbf{m}$ at any particular point in time t, it is possible to recover an approximate value of u at a previous point in time $t - \theta'$ for $0 \leq \theta' \leq \theta$:

$$\hat{u}(t-\theta') = \sum_{\ell=0}^{q-1} m_\ell d_\ell(\theta'), \qquad \text{where } d_\ell = \tilde{P}_\ell\left(\frac{\theta'}{\theta}\right), \tag{2}$$

where $\tilde{P}_\ell$ is the shifted Legendre polynomial of degree ℓ.

Another way to think of this system is that it encodes the past history of its inputs using a set of *temporal basis functions*. The particular temporal basis functions that are used are the Legendre polynomials, because they have been shown to be optimal for encoding such temporal memory (Voelker & Eliasmith, 2018). Of course, some information is lost in this process, and this is controlled by the dimensionality of the state variable $\mathbf{m}$, which is a q-dimensional vector. As q increases, more details (i.e., higher frequencies) about the past are kept in $\mathbf{m}$. The neural implementation of this computation is called a "Delay Network" (DN), and is also the core part of a novel machine learning algorithm known as the Legendre Memory Unit, which has been shown to outperform LSTMs on several benchmark tasks (Voelker, Kajić, & Eliasmith, 2019).

However, if we use biologically constrained neural networks to approximate this operation, then the actual computation performed by the system, and hence the quality of the time window encoded in $\mathbf{m}$, may be different. We can thus use the ideal computation expressed in eq. (1) as a benchmark.

In the following, we build various approximations of the DN using different biological constraints, systematically provide them with inputs, and evaluate their performance in terms of how well the input history can be recovered. In all cases, we compute the optimal recurrent and readout connection weights independently for each target population to approximate eq. (1) given the biological constraints. This avoids the need for a stochastic training process, such as backpropagation.

Neural Computation in the Cerebellum

The biological system of particular interest in this paper is the cerebellum. Not only is it well-studied and highly regular in its structure, but there are also reasons to believe that it does compute something akin to the operation performed by the Delay Network. Behaviourally, the cerebellum is known to be vital for some delay conditioning tasks, such as eye-blink conditioning (McCormick et al., 1981). In eye-blink conditioning, a sensory cue (e.g., an audio tone) is given before a puff of air into the eye. After some time, animals consistently learn to blink the correct amount of time after the sensory cue, such that the eye is closed when the puff actually happens.

There is no current consensus on how exactly the cerebellum learns these delays. One theory is that the responses observed in tasks such as eye-blink conditioning rely on intrinsic properties of the Purkinje cells (Lusk, Petter, MacDonald, & Meck, 2016). Another theory—and this is what we assume in this paper—is that the Granule-Golgi microcircuit is responsible for computing a temporal basis function representation (cf. Dean, Porrill, Ekerot, & Jörntell, 2010; Rössert, Dean, & Porrill, 2015), from which arbitrary delays can be linearly decoded. This is similar in principle to the Legendre basis functions used in the Delay Network. Indeed, Stöckel et al., 2020 show an initial implementation of eye-blink conditioning using this approach, but with less biological detail than is explored in this paper. We believe the findings in this paper provide another strong argument that the biology of the Granule-Golgi circuit is very well suited for implementing some kind of temporal basis function generation.

The Granule-Golgi microcircuit

The cerebellum contains about 50 billion granule cells, making them the most common type of neuron in the entire human brain. They are tiny cells that receive input from pre-cerebellar nuclei (PCN) via "mossy fibres" and project via "parallel

fibres" onto the Purkinje cells. Granule cells also have interneurons interspersed amongst them, known as Golgi cells, forming an inhibitory feedback loop with the granule cells. That is, granule cells excite Golgi cells, and Golgi cells inhibit granule cells (Ito, 2010). Notably, each granule cell on average receives input from only four mossy fibres, as well as one or two Golgi cells (Chadderton, Margrie, & Häusser, 2004). These numbers are known in the literature as the *convergence* of a projection. Furthermore, the connectivity between Golgi and granule cells is spatially constrained, i.e., Golgi cells only connect to granule cells in their vicinity. The ratio of granule to Golgi cells is about 430:1 (D'Angelo et al., 2013).

Levels of Biological Detail

To demonstrate our approach of adding biological detail, we present five models of the Granule-Golgi microcircuit of increasing complexity. While the first model is merely an abstract implementation of eq. (1), the final model respects spatial sparsity, convergence, tuning curves, and neurotransmitter constraints. All models are depicted in Fig. 2. In all cases, the scalar input u is received from one hundred spiking Leaky Integrate-and-Fire (LIF) neurons with randomly chosen tuning curves representing the PCN (see Model B for details).

Model A: "Direct" Implementation For this model, we directly solve the differential equation in eq. (1) by integration. That is, we have a single layer of "neurons" that are pure integrators (i.e., no non-linearity). The matrix $\mathbf{A}$ describes the recurrent connection weights, and $\mathbf{B}$ the input connection weights. This model does not distinguish between the granule and Golgi cells, and does not include details such as individual neurons or spikes. Instead, it focuses on the high-level theory of what the system is computing.

Model B: Single Population We now replace the integrators with a single layer of 200 spiking Leaky Integrate-and-Fire (LIF) neurons. These neurons form a distributed representation of $\mathbf{m}$ using a population code. Each neuron i is parametrized by a randomly chosen preferred stimulus vector $\mathbf{e}_i$ (for *encoder*), gain α_i and bias current J_i^{bias}, resulting in a desired response (i.e., tuning curve) for each neuron:

$$a_i(\mathbf{m}) = G[J_i(\mathbf{m})] = G[\alpha_i(\mathbf{e}_i \cdot \mathbf{m}) + J_i^{\text{bias}}], \qquad (3)$$

where G is neural response curve of the LIF neuron model. The parameters α and J_{bias} are randomly chosen from a distribution that ensures a maximum firing rate of 50 Hz to 100 Hz, consistent with biological recordings of granule cells (Chadderton et al., 2004). We then use least-squares to solve for optimal input and recurrent connection weights that result in these desired tuning curves while implementing the equivalent calculation as in Model A. Importantly, when solving for the recurrent connection weights, we also take into account the synaptic filter, which we model here as a decaying exponential (i.e., a low-pass). This is the standard process in the NEF (Eliasmith & Anderson, 2003).

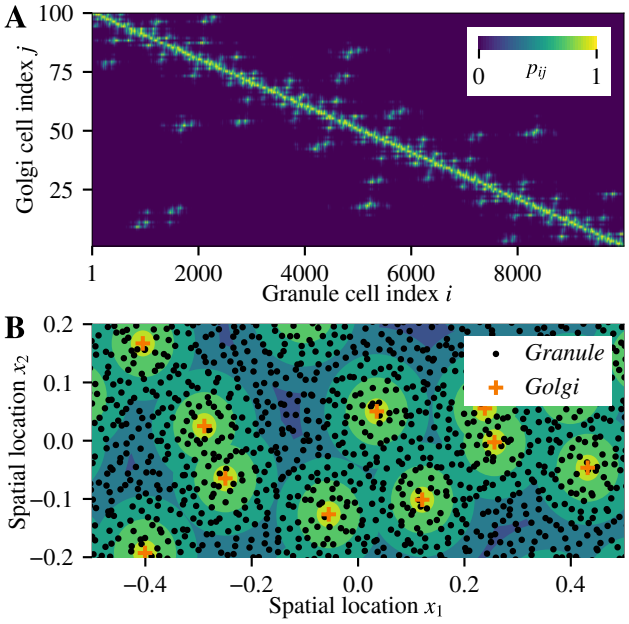


Figure 1: Spatial connectivity constraints. **(A)** Normalised connection probabilities p_{ij} for $\sigma = 0.25$. **(B)** Spatial organisation of the Golgi and granule cells. The background depicts the cumulative density of the Golgi to granule connection probability for a virtual Granule cell at each location (same colors as in **A**).

Model C: Inter-neurons As a next step, we separate the single layer of neurons into two separate populations corresponding to the Golgi and granule cells, reflecting the actual biology of the cerebellum (see above). This introduces two separate synaptic filters which need to be taken into account when solving for the connection weights that best approximate eq. (1). Furthermore, to at least approximate the fact that there are far fewer Golgi cells than granule cells, we use 20 Golgi cells and 200 granule cells.

Model D: Inhibition and Excitation So far, we have not accounted for Dale's principle, i.e., Golgi cells being purely inhibitory, and granule cells being purely excitatory. We handle this by switching to the non-negative least-squares problem described in Stöckel & Eliasmith, 2019. For each post-neuron i we minimize

$$\min_{\mathbf{w}_i^+, \mathbf{w}_i^-} \sum_{k=1}^{N} \left(\mathbf{w}_i^+ \cdot \mathbf{a}_k^+ - \mathbf{w}_i^- \cdot \mathbf{a}_k^- - J_i(\mathbf{m}_k)\right)^2 \text{ w.r.t. } \mathbf{w}_i^+, \mathbf{w}_i^- \geq 0,$$

where $\mathbf{a}_k^+$, $\mathbf{a}_k^-$ are the excitatory and inhibitory pre-activities for sample k, $\mathbf{w}^+$, $\mathbf{w}^-$ are the connection weights for excitatory and inhibitory pre-neurons, and $J_i(\mathbf{m}_k)$ is the current required to represent the desired value $\mathbf{m}_k$ as defined in eq. (3).

Model E: Sparse connectivity and activity For this model, we add in realistic constraints on how connected the neurons are. The previous models used all-to-all connections, whereas

for this model, we only allow a subset of those connections to be non-zero. This applies to both the input to the Golgi-Granule system and for the recurrent connections within the granule cells. In particular, we account for the granule cell convergence numbers by randomly selecting five PCN and five Golgi cells as pre-neurons—this number is slightly larger than the number reported above, since, as we discuss below, the number of pre-neurons places a strict upper limit on the connectivity (see below for more details). Given this extremely sparse connectivity, we increase the number of neurons in the simulation to 10 000 granule and one hundred Golgi cells, which is closer to the ratio observed in nature.

To account for spatially imposed connectivity constraints, we assign a location $\mathbf{x}$ in $[-1,1]^2$ to each neuron. The probability p_{ij} of a post-neuron i to receive a connection from a pre-neuron j is proportional to $\exp\left(-\|\mathbf{x}_i-\mathbf{x}_j\|^2/\sigma^2\right)$ (Fig. 1).

Finally, the input representation in the PCN cells was made more sparse by adjusting the tuning curves of the input neurons. Neural recordings indicate that there is very little input activity when no stimulus is present (Chadderton et al., 2004), while, per default, the randomly-chosen NEF tuning curves result in many neurons being active when representing the value $u = 0$.

Experiments and Results

To evaluate the effects of these biological details, we systematically generate two different types of input $u(t)$, feed those into the network and record the resulting activity. In particular, we present results with a periodic pulse input of varying pulse width t_{on} and band-limited white noise of varying bandwidth B. These are meant to depict typical sorts of inputs that may arise in experimental situations (pulses) and more real-world situations (band-limited white noise). We then determine how accurately the past history of u over the window θ can be recovered from the resulting network activity via optimal linear readout weights. We use $\theta = 0.4\,\text{s}$ in all simulations; each individual experiment simulates the network for $T = 10\,\text{s}$. The error is measured as the RMSE of the reconstruction divided by the RMS power of the input signal itself (normalized RMSE, or NRMSE).

The overall results for all five models are shown in Fig. 3. This shows the average reconstruction error for varying inputs (horizontal axis) and for varying time delays (vertical axis) over ten trials. An example run of Model E (the model with the most biological detail) is shown in Fig. 4. The different decoded output lines (bottom) are all based on the neural activity (middle), but with different readout weights. These approximate the input value u with various time delays over the entire time window, from the immediate input right now ($\theta'/\theta = 0$) to θ seconds ago ($\theta'/\theta = 1$).

As can be seen in Fig. 3, the network successfully functions as a method for encoding the temporal pattern of input data over the desired window of time θ. Adding more biological detail decreases the accuracy with which the system approximates eq. (1), but most of the information is still encoded. The input pulses (Fig. 4A) show that the reconstruction is a smoother version of the input; the system is not good at representing sudden changes, and this is the main source of noise in the reconstruction. This is as expected from using smooth Legendre polynomials as temporal basis functions.

Furthermore, we note that there is a peak in accuracy when decoding data from $\theta' = 70\,\text{ms}$ in the past ($\theta'/\theta = 0.175$), and this peak is more pronounced as more biological detail is added. This corresponds to the neurotransmitter time constant $\tau = 70\,\text{ms}$ we use for all connections, and which is based on a first-order low-pass fit to the Granule-Golgi dynamics reported in Dieudonné (1998). Importantly, we can use the model to determine what the accuracy would be like if we changed this value. This is shown in Fig. 5A. Interestingly, the performance of the system is best in the range between 50 ms to 60 ms, which is relatively close to what we observe in nature. Both smaller and larger values lead to an increase in error.

As discussed above, a striking feature of the cerebellar microcircuitry is the low granule cell convergence. One possible hypothesis is that these numbers are a trade-off between minimizing connectivity and the overall performance of the resulting system. In our model, we can test this hypothesis by systematically varying the number of pre-neurons the solver has access to. Results are shown in Figs. 5B and C. The performance of the system does improve with larger limits, yet plateaus at still relatively small convergences. More importantly, as mentioned above, the specified desired convergence solely controls the number of *potential* pre-neurons. Since the neural weight solver may still set a weight to zero, these convergence numbers are strict upper limits. Measuring the actual convergence in the PCN to granule connections (Fig. 5D), we see a peak at one to three PCN neurons connecting to each granule cell. This peak is almost independent of the desired convergence number and close to observed averages.

Discussion

We successfully mapped a high-level, mathematical function onto a brain microcircuit while incorporating biological constraints. This process was simplified by the ability of our modeling tool to automatically account for Dale's principle, spatial constraints, as well as convergence numbers.

Our results show that the Granule-Golgi circuit could in principle implement a temporal basis function representation, which is in agreement with existing hypotheses about cerebellar function. Measurements from our model could be used to generate hypotheses about the kind of electrophysiological data we would expect to find, if this function was indeed realised in the brain. Having access to low-level biological parameters *in silico* furthermore facilitates the exploration of physiological changes that are difficult to achieve experimentally *in vivo*. As discussed above with respect the to synaptic time constants and convergence numbers, this allows us to investigate why certain parameters are as observed.

A key difference of our approach to existing models of the Granule-Golgi circuit (such as Rössert et al., 2015), is that

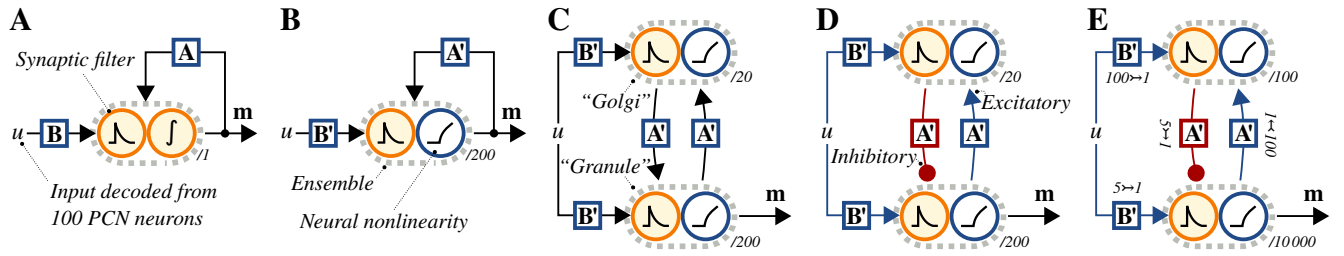


Figure 2: Network types used in our experiments. **(A)** "Direct" implementation with an optimal integrator. **(B)** Using the synaptic filter of a single population of spiking neurons for temporal integration. **(C)** Inter-neuron population in the recurrent path. **(D)** Same as **C**, but accounting for Dale's principle. **(E)** Same as **D**, but with more detailed biological constraints (see text).

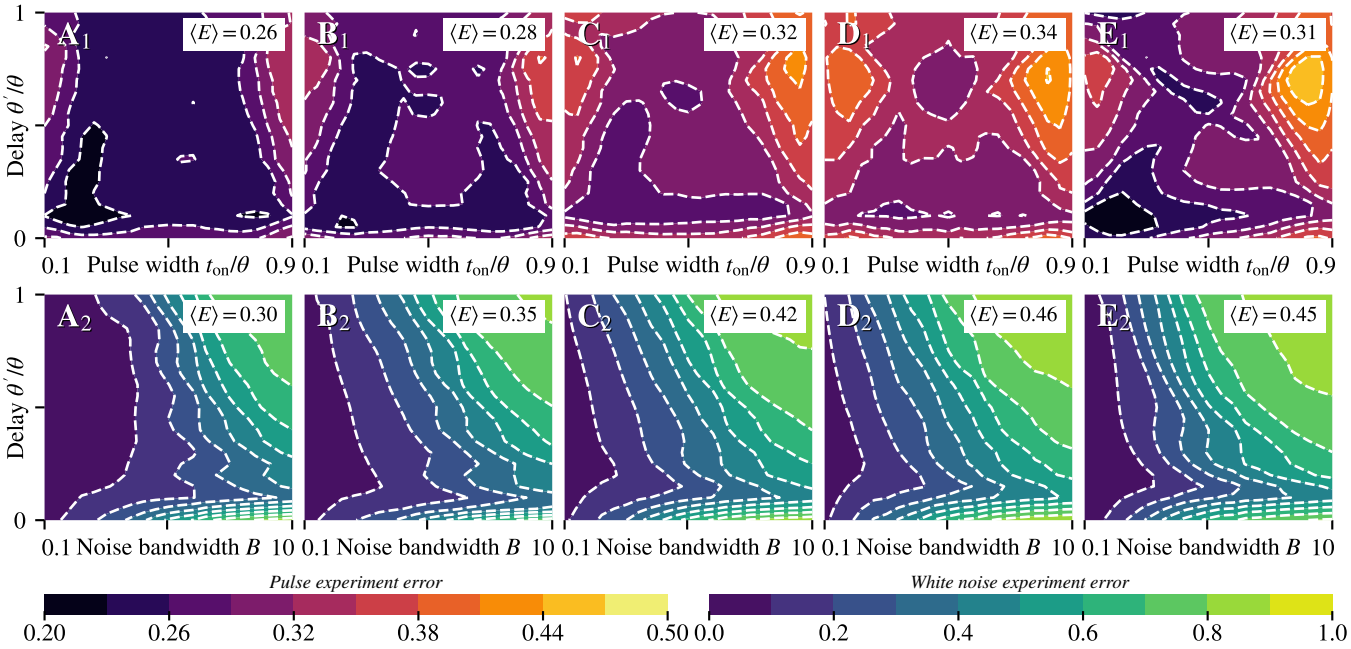


Figure 3: Delayed signal reconstruction error for different types of input signals, delays, and network types. All error values are expressed as RMSE divided by the average RMS of the input signal over ten trials. Columns correspond to the network types in Fig. 2 above. *Top row:* Reconstruction error for rectangle pulse signals of varying width. *Bottom row:* Reconstruction error for a band-limited white noise input signal with varying band-limit.

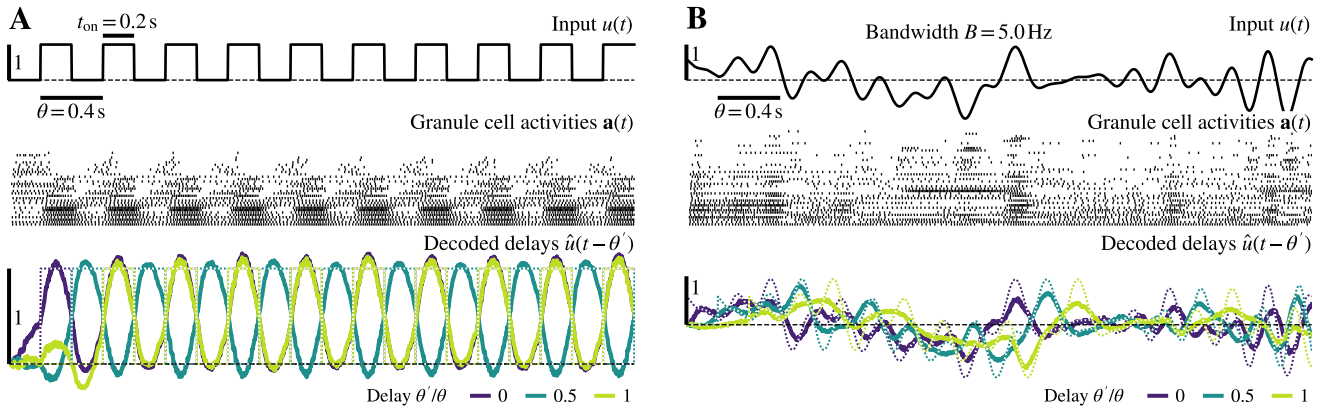


Figure 4: Examples showing the delayed input signals decoded form the granule layer in the detailed model (Fig. 2E). *Top row:* Input signal (rectangle pulses in **A**, white noise in **B**). *Middle row:* Spike raster for 40 randomly selected granule cells. *Bottom row:* Delays decoded from one thousand randomly selected granule cells. Dotted lines correspond to an optimal delay.

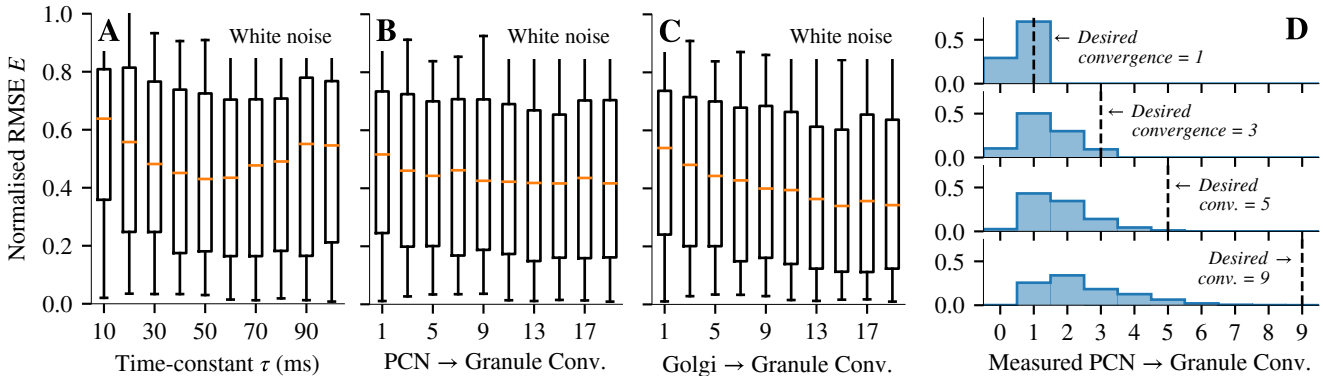


Figure 5: Parameter exploration. **(A-C)** Effects of varying parameters on the delay reconstruction error. The box plots show the median, lower and upper quartile of all the data depicted as a contour plot in Fig. 3; whiskers are the min/max. The total number of data-points in each bar plot is $n = 441$. **(D)** Histograms showing the frequency of measured PCN to granule convergences.

our modeling techniques are more general with respect to the high-level function that is being mapped onto the underlying circuit. Instead of relying on random connectivity, we directly specify the high-level function we would like the system to perform. We encourage cognitive modellers to view this particular model as an example; the techniques we present here are in principle compatible with all NEF models, including models of cognitive phenomena using the Semantic Pointer Architecture (SPA; Eliasmith, 2013).

We hope that this research facilitates grounding cognitive theories in biological mechanisms beyond what was already possible with the NEF and Nengo. Future work will focus on incorporating additional biological detail into the model (such as separate biological time constants for all synapses), as well as applying our techniques to more complex models.

References

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., ... Eliasmith, C. (2014). Nengo: A python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, *7*(48).

Chadderton, P., Margrie, T. W., & Häusser, M. (2004). Integration of quanta in cerebellar granule cells during sensory processing. *Nature*, *428*(6985), 856–860.

Dean, P., Porrill, J., Ekerot, C.-F., & Jörntell, H. (2010). The cerebellar microcircuit as an adaptive filter: Experimental and computational evidence. *Nat. Rev. Neurosci.*, *11*(1).

Dieudonné, S. (1998). Submillisecond kinetics and low efficacy of parallel fibre-Golgi cell synaptic currents in the rat cerebellum. *The Journal of Physiology*, *510*(3).

D'Angelo, E., Solinas, S., Mapelli, J., Gandolfi, D., Mapelli, L., & Prestori, F. (2013). The cerebellar Golgi cell and spatiotemporal organization of granular layer activity. *Frontiers in Neural Circuits*, *7*, 93.

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford University Press.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT Press.

Eliasmith, C., & Kolbeck, C. (2015). Marr's Attacks: On Reductionism and Vagueness. *TopiCS*, 1–13.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Ito, M. (2010). Cerebellar Cortex. In G. Shepherd & S. Grillner (Eds.), *Handbook of Brain Microcircuits* (1st ed., pp. 293–300). Oxford University Press.

Lusk, N. A., Petter, E. A., MacDonald, C. J., & Meck, W. H. (2016). Cerebellar, hippocampal, and striatal time cells. *Current Opinion in Behavioral Sciences*, *8*.

Marr, D., & Poggio, T. (1976). From Understanding Computation to Understanding Neural Circuitry. *MIT AI Memo*(357).

McCormick, D. A., Lavond, D. G., Clark, G. A., Kettner, R. E., Rising, C. E., & Thompson, R. F. (1981). The engram found? Role of the cerebellum in classical conditioning of nictitating membrane and eyelid responses. *Bulletin of the Psychonomic Society*, *18*(3).

Parisien, C., Anderson, C. H., & Eliasmith, C. (2008). Solving the problem of negative synaptic weights in cortical models. *Neural Computation*, *20*, 1473–1494.

Rössert, C., Dean, P., & Porrill, J. (2015). At the edge of chaos: How cerebellar granular layer network dynamics can provide the basis for temporal filters. *PLOS Computational Biology*, *11*(10).

Stöckel, A., Stewart, T. C., & Eliasmith, C. (2020). A biologically plausible spiking neural model of eyeblink conditioning in the cerebellum. In *Proceedings of the 42th Annual Conference of the Cognitive Science Society.* Cognitive Science Society.

Stöckel, A., & Eliasmith, C. (2019). Passive nonlinear dendritic interactions as a general computational resource in functional spiking neural networks. *arXiv*.

Voelker, A. R., & Eliasmith, C. (2018). Improving Spiking Dynamical Networks: Accurate Delays, Higher-Order Synapses, and Time Cells. *Neural Computation*, *30*(3).

Voelker, A. R., Kajić, I., & Eliasmith, C. (2019). Legendre memory units: Continuous-time representation in recurrent neural networks. In *Advances in NeurIPS*.

Characterizing Pause Behaviors in a Science Inquiry Task

Caitlin Tenison and Burcu Arslan
(ctenison@ets.org) (barslan@ets.org)
Cognitive and Technology Sciences Center
Educational Testing Service, Princeton, NJ, 08541, USA

Abstract

In inquiry-based learning tasks students are actively involved in learning knowledge and skills through experimentation. The success of these activities largely depends on student's inquiry practices. While traditional assessment infers student competency from their responses and problem-solving steps, the pauses between these actions provide a valuable source of information. Pauses during inquiry tasks capture a wide range of productive and unproductive activities such as planning, reasoning and mind-wandering. We present efforts to characterize the pause behaviors during a science inquiry task using hidden Markov modeling. We explore how theory can inform data driven modeling approaches, describe initial evidence of meaningful pause states, and consider the limitations of this approach for supporting inferences about students' science inquiry practices.

Keywords: Science inquiry, Pauses, Process modeling, Hidden Markov modeling.

Introduction

Several probabilistic and data mining approaches have been used to infer student knowledge and skills from process data (Levy & Mislevy, 2016). Most of these approaches focus on the correctness of the steps that students perform (Yudelson, Koedinger & Gordon, 2013). While effective in domains where correctness is clearly defined, these approaches have limited application in inquiry-based interactive tasks in which students discover and apply relevant knowledge and skills, form and test hypotheses, and then reflect on the outcome of those tests. For these tasks, considering the problem-solving process and the cognitive mechanisms underlying those actions are important when making valid inferences about students' science inquiry practices. While we can draw inferences about the problem-solving process through modeling the actions students take, modeling the pauses between actions can also support inferences about underlying cognitive processes supporting those actions. The goal of our current research is to identify methods for characterizing pauses in the problem-solving process and establish what these pauses contribute to the measurement of inquiry skill. We explore how theory can inform data driven approaches and describe initial evidence, while weighing the limitations of this approach for supporting inferences about students' science inquiry practices.

Evaluating Pauses in Educational Activities

Early studies on pause behaviors during problem solving indicate that pauses between actions provide insight into the processes supporting task completion (e.g., VanLehn, 1991). Given the variability in the number and length of pauses between students, there have been several attempts to use this information to characterize the proficiency of the student (Pelanek, 2014; Dang, Yudelson, & Koedinger, 2017). In an expansion of Bayesian Knowledge Tracing (BKT), Pelanek (2014) incorporates timing into the approach to improve estimates of skill level of students using a tutoring system. In this work longer pauses decrease the probability that the student has mastered the skill. This assumption may be correct when fluency is the goal of learning but is less appropriate when pausing reflects productive behaviors. For example, in their model of diligence, Dang, Yudelson, and Koedinger (2017) propose the use of both time and performance to separate the impact of productive and unproductive pauses on their measure. Other studies take a more unstructured approach and use pauses as input for machine learning algorithms predicting different constructs such as carefulness (Banawan, Andres, & Rodrigo, 2017). A similarity across all these approaches is that they aggregate pauses within the problem-solving process to create general measures of time on task rather than considering the occurrence of these pauses within the process data.

Pauses at different points throughout an educational activity are indicative of different cognitive activities. Prior research using pauses to identify periods of wheel-spinning, gaming the tutor and productive persistence rely on expert qualitative coding and the structure of the tutor environment to identify when a pause is likely to indicate these behaviors (Paquette et al., 2014; Aleven et al., 2004). While these pauses are informative for modeling student behavior, they require human coding and, for this reason, tend to be used to characterize pauses in tutors with a limited space of actions.

Inquiry Learning Activities

Inquiry-based learning environments support students in learning concepts through the exploration and development of general learning behaviors and strategies. Inquiry learning activities are popular in science education as a means of teaching scientific principles through the application of the scientific method. Interactive science simulation environments require students to interact in an open-ended task to generate responses that help support the collection of evidence about what students know (declarative knowledge) and can do (procedural knowledge). Pauses within inquiry learning environments can reflect a blend of productive and off-task behaviors. The nature of these pauses can be inferred by using the process data to understand the context in which the pause occurs. However, within open-ended tasks this type of inference is non-trivial because students can produce a wide range of distinct actions and the underlying cognitive processes of these actions are not directly observable (Ercikan & Pelligrino, 2017).

Prior attempts to model student pause behavior in scientific inquiry activities include both data-driven approaches for identifying behavioral patterns and theory-based approaches that model the reasoning and learning involved in scientific inquiry-based tasks. In their work studying student experimentation with an electrical circuit simulation, Perez and colleagues (2017) used coded log data to compare sequences. They found no difference in overall pause frequency across low and high performing students; however, high performing students paused more than low performing students before and after running experiments. This method requires careful coding of the log data and assumes no measurement error, which can be problematic in situations where there is both variability and the potential for interruptions in strategy execution. Theory based approaches have the flexibility to capture the variation in behavior within a simulation environment. In their Simulated Psychology Lab (SPL) task, Schunn and Anderson (1998), created an environment where people could design research studies, collect simulated data and manipulate that data to draw conclusions. Using the SPL environment Schunn and Anderson compared the actions of human subjects to actions of their cognitive model of scientific discovery. This SPL model was able to capture variation in scientific inquiry behaviors due to the structure of the environment and prior experience. While the SPL model was not used to predict timing data, it provides insight into the activity that occurs during the pauses between actions in a scientific inquiry task.

Current Study

In the current study, we model the pause behaviors of students as they interact with a science simulation task designed to assess students' science inquiry practices in relation to the concept of saturation (i.e., maximum concentration) and control-of-variable strategy (Figure 1). This environment provides students with the tools to run experiments, organize data, and report conclusions. The interactive nature of this environment provides students with considerable variability in how they complete the task(e.g., number of trials, strategy use, timing of actions). This variability while likely capturing meaningful differences in science inquiry ability also makes comparison across students challenging.

We use hidden Markov models (HMMs) as an exploratory technique to distinguish different pausing behaviors as they occur in the context of student actions. HMMs capture the probabilistic transition between latent states in sequential time steps. The strength of these models is the Markovian assumption that the probability of the current state is driven by the previous state and the currently observed behavior. HMMs have been used in the educational data mining community to identify behavioral patterns that can be linked to meaningful cognitive states (e.g., BKT) and strategies (Tenison & MacLellan, 2014). We are not aware of any prior work using HMMs to model pause behaviors within student process data from adaptive learning and assessment environments.

The focus of our modeling effort is to characterize student's pause behavior in a science inquiry task. We hypothesize that the pauses observed during inquiry represent distinct cognitive activities. We further propose that optimal characterization of the processing occurring during pauses will provide useful information to improve our assessment of knowledge skills and abilities of students. We use HMMs to address the challenge of characterizing pauses from individual process data. We use theory to guide our construction of these models but allow data to drive the models we fit. To evaluate the descriptiveness of the model, we use three metrics: 1) sensitivity to group differences, 2) prediction of correct conclusions from patterns of pauses during the scientific inquiry activity, and 3) agreement with validated external measures of scientific inquiry skills.

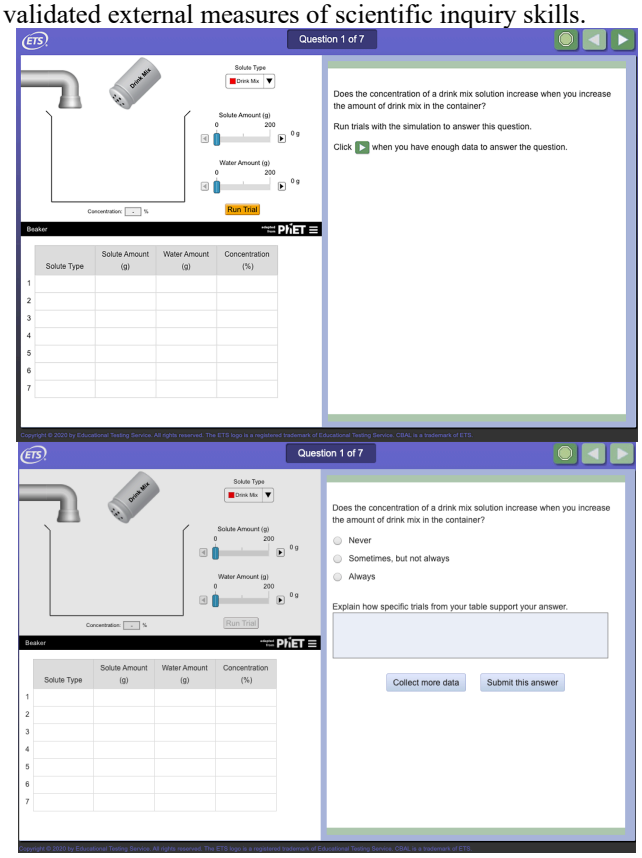


Figure 1. A screenshot of question 1 in the concentration simulation: a) Science inquiry screen, b) Answer screen.

Methods

Participants

Two-hundred-seventy-three students in 6th and 8th grades were tested with a concentration simulation task. Six students were excluded from the analysis due to process data problems. The final data included 134, 6th grade students (N_{Female}= 72) and 131, 8th grade students (N_{Female}= 67). Two students had no grade information associated with their data.

Materials

The concentration simulation task we used was originally developed by PhET (Wieman et al., 2008) and modified for the purpose of the study to include selected and constructed response questions (see Finn, 2018 for a more detailed task description). The simulation was an HTML5 application written in JavaScript and delivered through a standard web browser. In total, there were 7 questions in the task. In the scope of this study, we focus on students' science inquiry behavior (i.e., observable actions and pauses between actions) as students complete the first question within the task, together with, their submitted answers to that question.

This question asks students to run experiments to investigate whether the concentration of a solution increases when the amount of solute increases. Students were instructed to click the next button when they had enough data to answer the question (see Figure 1a). The open-ended nature of the task allowed students to prepare and run experiments by: (i) running as many trials as necessary to give an answer, (ii) setting any value between 0 and 200g for both solute and water amounts, (iii) using different strategies during investigation (e.g., control-of-variable; varying, increasing, or decreasing both variables at the same time). For each student run simulation, results are updated in a 'data-table'. Students can manipulate this data by reordering or deleting experimental records from their data-table. While interacting with the data-table is not required, it is meant as a workspace for students to organize the results of their investigations when drawing conclusions about properties of the concentration solution. After clicking the next button, the answer options for the question together with a constructed response box asking students to justify their selected response appeared on the second screen (see Figure 1b).

Process Data Representation

The representation of the student's scientific inquiry activities that we use to fit our model impacts the descriptiveness of that model. The raw data logfiles the system produced recorded detailed information about specific student actions and system events. From these logs, we identified 10 general categories that capture the actions corresponding to the subcomponents of the concentration simulation task (Table 1). We chose these categories to align with the top-level goal structure of Schunn and Anderson's SPL model (1998). For simplicity, we refer to these categories as '*Actions'* but use labels that indicate whether or not these actions were produced by the student or the interface.

Actions generated by the simulation environment introduce new information to the student. These actions represent standard instructions along with updates to the data collection table based on the experiments that students run. Student actions are changes made to the environment in response to the information the environment provides them. These actions unfold over time. Actions are separated by variable periods where no activity is logged. We refer to these periods as '*Pauses'*. In our analysis, we are interested in characterizing the types of pauses that reflect periods of inactivity that are within the control of the student. We do not analyze the pauses that reflect the time it takes for the simulation environment to run the simulation (e.g., the pauses between the start and end of the animation).

Table 1: Average Percentage (SE) of Session Comprised by Actions types and a Sample Action Sequence.

Representation of Actions	**Percentage**
Experiment Preparation	31.8 (1.2)
Experiment Run	8.6 (.23)
Table Manipulation	4.1 (.5)
Answering Questions	6.7 (.43)
Collect More Data	0.63 (.06)
Done	2.3 (.15)
Interface: Error Message	0.45 (.12)
Interface: Load First Page	2.3 (.15)
Interface: Load Second Page	2.6 (.13)
Interface: Update Table	8.6 (.23)
Sample Action Sequence:	
Interface_Load_FirstPage -> Long_Pause -> Experiment_Prep -> Med_Pause -> Experiment_Prep -> Experiment_Prep -> …	

The action sequence representation (Table 1, bottom row) captures the sequence in which actions occur: however, it does not capture specific temporal information about when these events occur. Prior work using pauses to model cognitive processing and proficiency suggests that the length of the pause is an especially important indicator of what is occurring during that period (Paquette et al., 2014). We explored several different methods of representing the data to capture differences in pause length in HMMs. We considered representing actions as a binned timeseries: however, under this representation pauses dominated the sequences and previous research has indicated HMMs are sensitive to over dispersion (Olteanu & Ridgway, 2012).

Instead, we chose to assign pauses to ordinal categories based on length. Pauses less than 250 ms were ignored because on average motor preparation takes 250 ms (Anderson, 2007) and if an action is preceded by such a pause, it was unlikely to reflect meaningful cognitive activity. For pauses longer than 250 ms, we used the 25th and 75th percentiles as cut-points to categorize pause durations (Figure 2). Pauses between 250 ms and 1.3s were labeled "Short Pauses", pauses between 1.3s and 6.2s seconds were considered "Medium Pauses", and pauses greater than 6.2s seconds were labeled "Long Pauses". In Figure 2, we illustrate the distribution of pauses longer than 250 ms across all sequences with vertical lines indicating category cut-offs. On average, pauses lasted 6.4 s (SD = 12.8). Pauses accounted for 27.4% of the action sequences described in Table 1 (Short: 7.2% (.38), Medium: 16.6% (.45) and Long: 8.2% (.23)).

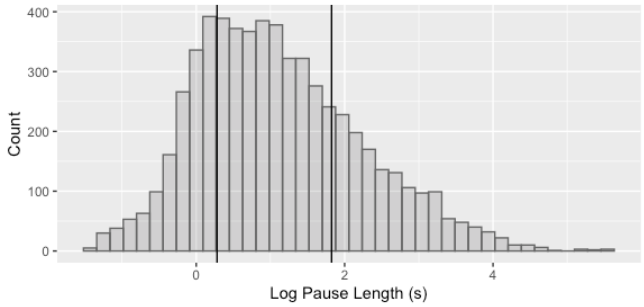


Figure 2: Histogram of the duration of pauses (log-seconds). Vertical lines indicate cut-points.

Hidden Markov Models

Our aim in fitting the HMM to the student's action sequences is to identify hidden states that help us characterize distinct categories of pauses from the context in which they appear and provide us with useful descriptive information about how pauses are expressed in the problem-solving process. We used the R package seqHMM (Helske & Helske, 2017) to fit our HMM models. We use random priors to initialize our emission and transition probabilities, with the exception of the done state which was given a 0 probability of the model starting in that state and a 0 probability of transitioning from that state to any other state. This means the model has no expectation about the types of states present nor any expectation about how people might be moving between these states. For each model we ran the EM algorithm 10 times with randomized starting values for the transition and emission probabilities to avoid fitting local optima.

Results

Models of Pausing Behavior

We fit HMMs with between 3 and 25 states to the data. We used Bayesian Information Criteria (BIC) to determine which model best fit the data while also penalizing for added parameters to avoid overfitting (Figure 3), lower values indicate better model fits. We found that a 16-state model best fit the data (BIC 40504.3, log likelihood -17,6661.5). The next best fitting model (18 state) has a BIC 544.6 points higher than the 16-state model. Typically, a BIC difference greater than 10 is considered strong evidence against the higher BIC value.

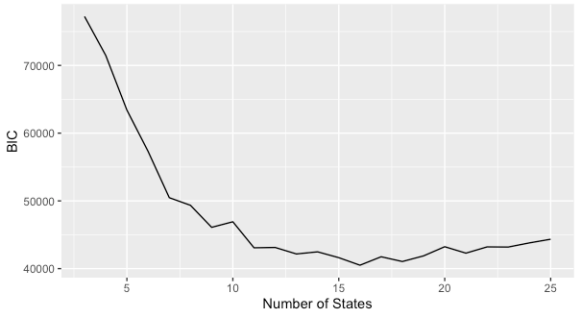


Figure 3: Model fit (BIC) for 3 through 25 state models.

Interpreting Hidden States

We illustrate the structure of the best fitting HMM in Figure 4. The nodes represent the hidden states, the colors of the nodes reflect the probability of that hidden state emitting action events (color coded in the legend of Figure 4). Hidden states emitting pauses show the greatest division across different observable actions. The arrows between nodes represent the transition probabilities, with labels and density reflecting specific probability. For readability we grouped related action states and do not display transition probabilities less than .05.

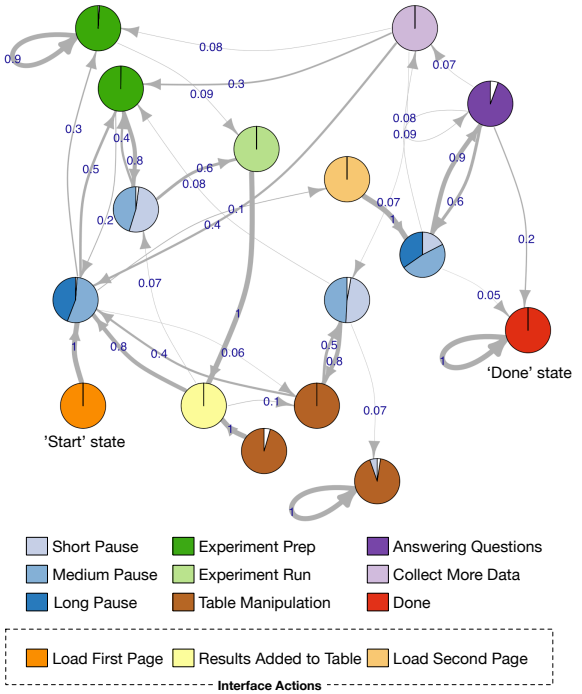


Figure 4: 16-State HMM.

Action states. We can distinguish the states our HMM fits into two categories, *action states* and *pause states*. Our 16-State model fits several states that have a high probability of emitting the same action but fit very distinct transition probability profiles. All of these states had a probability greater than or equal to 95% of emitting to a single state. The only exception, a table-manipulation state, has a 7.6% probability of also emitting a short pause. While these hidden states are likely to emit a single action, the model does make an interesting separation of experiment-preparation and table-manipulation states. For both types of actions the model separates action states that have a high transition probability back into themselves and action states that have a high transition probability into pause states. This distinguishes actions that are swiftly executed as part of a sequence as opposed to actions separated by pauses. This distinction of 'swift' and 'thoughtful' actions could reflect differences in when planning or strategy selection occurs (before or during

task execution), or various maladaptive behaviors (e.g. wheel-spinning, mind-wandering).

Pause states. Our primary interest in this work is to explore whether this modeling approach can be used to separate pause states that represent periods of distinct cognitive processing. Our model fit four distinct pause states. We characterize these pauses as they appear in Figure 4 from left to right:

1. **Processing new information and goal setting:** This state captures the pauses that occur immediately after the task starts (96%), and after simulation results are added to the data-collection table (80%). Other states with over a 40% chance of transitioning into this state include a table-manipulation state and the state capturing the decision to run more experiments. From this pause state, students have a high probability of transitioning into an experimental preparation state (75%) or deciding they have collected enough data to progress with the task (14% probability of loading 2nd page). This state consists primarily of medium and long pauses, and an analysis of pause durations of this state indicates an average pause length of 10.4s (SE = 3.9). Given the transition probabilities and length of these pauses, we hypothesize that this state captures an amalgam of processing of new information, deciding to run an experiment or continue to the second page, and planning the experiment.
2. **Experimental Investigation:** This pause state connects to action states involved in preparing and running experiments. These pauses however are relatively short (M = 1.6s, SE = 0.43s). This state may distinguish thoughtful (or aimless) experimental preparation from swift experimental preparation.
3. **Data Manipulation:** This pause state has a high probability of transitioning to (80%) and from (54%) one of the table-manipulation states. As with experiment running this distinguishes thoughtful and swift table-manipulation states. Pauses in this state are relatively short (M = 2.0s, SE = 0.7).
4. **Reflection on Questions:** Our final pause state connects activities on the second part of the science inquiry task (question answering, finishing, and deciding to collect more data). The longer length of these pauses (M = 8.6s, SE = 5.2s) likely capture the time participants spend encoding the answer options, reading and reflecting on responses to the two questions on this page.

Our HMM distinguishes between the scientific inquiry and question answering activities as well as between planning activities, which take longer and capture the switch between high-level task goals and task execution decisions that tend to be much shorter and distinguish thoughtful and swift actions. While this information provides us with a compelling descriptive account of how students complete this inquiry task, we are still limited in our ability to infer the specific cognitive processes that occur during these pause states.

The Role of Pauses in Scientific Inquiry

Our goal in characterizing the different pause states present within the process data of students is to use this information to improve our assessment of scientific inquiry. In our first step to determine if this behavior characterization is informative, we consider 1) whether we see differences in pause behavior between our 6th and 8th grade cohorts, 2) whether the occurrence of different types of pauses are predictive of students answering the question correctly, and 3) whether pauses explain variance in validated measures of scientific inquiry.

We compared the relative proportions of activity spent in the four different pause states of 6th and 8th graders in our sample. We used an independent 2-group Mann-Whitney U test to account for non-normality of the data. Using this test, we found no significant differences in the proportion of activity spent in the four pause states, other than a marginally significant difference in the Data Manipulation Pause state ($W = 9648.5$, $p = .055$), which was more frequently observed in 6th graders compared (1.6%) with 8th graders (.9%).

The scientific inquiry activity we modeled includes a multiple-choice question that asked students to judge whether the concentration of the mixture always increases. This question measures student understanding of the principle of saturation which could result from either their inquiry practices or prior domain knowledge. We tested whether the proportion of the time spent in the different pause states during the inquiry activity were predictive of whether participants got the answer of the multiple-choice question correct. Using a logistic regression, we included all four pause states as factors to predict score on that item. The overall variance explained by this model is low (McFadden pseudo $R^2 = .02$). Two of the pause states were marginally significant: Information Processing (OR: .003, $\beta = -5.8$, $z = -1.7$, $p = .086$) and Experimental Investigation (OR: .25, $\beta = 3.3$, $z = 1.8$, $p = .078$). These low odds ratios indicate students with greater pause activity are more likely to draw the wrong conclusion.

Prior to interacting with the task, students were administered the Waves Benchmark Assessment (WBA) as part of the SimScientists assessment suite developed by WestEd (Quellmalz, Timms, & Buckley, 2010). This measure uses a different science domain but attempts to evaluate the inquiry skills of students. As a first step in determining if the pause states identified by our HMM could provide information about the inquiry skills of the student, we look at the concordance between the pause behaviors and the WBA measure. We fit a linear regression model to measure how much of the variance in the WBA measure we could capture using the proportion of student actions within the four HMM pause states. We started with a maximal model with main effects for all four states and performed stepwise model selection using Akaike information criteria (AIC) to compare fits. Our final model indicated a significant collective effect of the Experimental Investigation, Data Manipulation, and Reflection on Questions pause states ($F(3,188) = 7.6$, $p < .001$, adjusted-$R^2 = .09$, BIC = 1304.6). To test if the HMM states account for more variance in the WBA measure than the raw pause actions, we fit a separate linear regression using the proportion of short, medium, and long

pauses in a student's action sequence to predict the WBA measure. Using stepwise AIC model selection, we found that the maximal model best fit the data ($F(3,188)=4.8$, $p < .005$, adjusted-$R^2 = .05$, BIC=1312.3). This model does not fit the data as well as, nor explain as much of the variance in the WBA score as the pause states model.

Discussion

In this paper, we show how an unsupervised modeling approach can be used to characterize pauses in the problem-solving process and explored what these pauses contribute to the measurement of skill. Our best fitting model identified four distinct pause states and split action states around experimental preparation and table manipulation activities into separate swift and thoughtful action states. These results suggest that pauses capture a range of processes and aggregation across pauses obscures meaningful variation in students' inquiry practices. We found weak evidence that 6th graders and 8th graders pause with similar frequency, but 6th graders pause slightly more when manipulating data. Across grades, students who paused while setting up experiments and in between inquiry activities were score incorrectly on the subsequent multiple-choice item. Finally, we found that pauses around experimental preparation, data manipulation and question answering varied in concordance with student's science inquiry ability, and that our characterization of pauses explained 4% more variance than considering only pause length. The finding that pauses explain a small proportion of variance in scores on the multiple-choice item and the WBA measure illustrates the challenge of using this information to evaluate the inquiry process. The conclusions students draw only partially reflect inquiry ability and future work validating models of inquiry process would benefit from more direct measures of planning, investigation, and analysis skills.

It is unlikely HMMs can discover the structure of problem-solving strategy at the same granularity of cognitive architectures (e.g., ACT-R; Anderson, 2007); however, these models can be used to provide a computational formulation of behavior patterns that balances an adherence to cognitive science theory, parsimony and conformity to data. Our goal in fitting a descriptive model was to better understand differences between individuals and capture meaningful variation in skill. While we examine group differences in a post-hoc analyses, in future research this information could be used as covariates within our HMM to guide model fitting (Helske & Helske, 2017). Such an approach would be especially appropriate in situations where there are clear hypotheses about how factors such as student ability and group membership drive differences in behavior patterns. The results of such models can be used to focus the subtasks we construct cognitive models to capture.

The precision of HMMs in capturing cognitive activity is limited by our ability to observe the thinking of the student. This is especially obvious in our first pause state which combines processing new information, deciding the next action and planning the execution of that action. In future work, we plan on exploring several avenues for improving our ability to distinguish pauses. In the current study, we found that how we represented pauses impacted the descriptiveness of our model. One approach for improving the distinction between pause states is to extend our representation of the actions students take to include more information about the actions. Including information about the data collection strategy students use may help us to identify pause behavior related to specific experiment goal subtypes such as those identified in the SPL model (Schunn & Anderson, 1998). We believe the combination of data and theory within these models will lead to promising avenues for assessing inquiry skills.

Acknowledgments

This work was funded by Educational Testing Service.

References

Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2004, August). Toward tutoring help seeking. In *International Conference on Intelligent Tutoring Systems* (pp. 227-239).

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford: Oxford University Press.

Banawan, M. P., Andres, J. L., & Rodrigo, M. T. (2017, October). Predicting student carefulness in an educational game for physics using semi-supervised learning. In *Conference on Information Technology Education* (19-21).

Dang, S., Yudelson, M., & Koedinger, K. R. (2017, April). Detecting diligence with online behaviors on intelligent tutoring systems. In Proc. *ACM Conference on Learning@ Scale* (51-59).

Ercikan, K., & Pellegrino, J. W. (2017). *Validation of Score Meaning for the Next Generation of Assessments: The Use of Response Processes*. New York, NY: Routledge.

Finn, B. (2018). Metacognitive evaluations during science simulations: How do ratings of confidence and understanding relate to science assessment inquiry processes? *Archives of Scientific Psychology*, 6(1),117-129.

Helske, S., & Helske, J. (2017). Mixture hidden Markov models for sequence data: The seqHMM package in R. arXiv preprint:1704.00543.

Levy, R., & Mislevy, R. J. (2016). *Bayesian Psychometric Modeling*. Boca Raton, FL: CRC Press.

Paquette, L., de Carvalho, A. M., & Baker, R. S. (2014, July). Towards understanding expert coding of student disengagement in online learning. In Proc. *Cog. Sci. Soc.*

Pelánek, R. (2014). Application of time decay functions and the Elo system in student modeling. In Proc. *Educational Data Mining*.

Perez, S., Massey-Allard, J., Butler, D., Ives, J., Bonn, D., Yee, N., & Roll, I. (2017). Identifying productive inquiry in virtual labs using sequence mining. In Proc. *International Conference on Artificial Intelligence in Education* (287-298).

Quellmalz, E. S., Timms, M. J., & Buckley, B. (2010). The promise of simulation-based science assessment: The

Calipers project. In Proc. *International Journal of Learning Technology*, 5(3), 243-263.

Schunn, C. D. & Anderson, J. R. (1998). Scientific discovery. In J. R. Anderson, & C. Lebiere (Eds.). *The Atomic Components of Thought*, 385-428.

Tenison, C., & MacLellan, C. (2014). Modeling strategy use in an intelligent tutoring system: implications for strategic flexibility. In Proc. *International Conference on Intelligent Tutoring Systems* (pp. 466-475).

Wieman, C., Adams, W., & Perkins, K. (2008). PhET: Simulations that enhance learning. *Science,* 322, 682–683.

VanLehn (1991). Rule acquisition events in the discovery of problem-solving strategies. *Cognitive Science*, 15(1), 1-47.

Yudelson, M., Koedinger, K., Gordon, G. (2013) Individualized Bayesian knowledge tracing models. In Proc. *International Conference on Artificial Intelligence in Education* (pp. 171-180).

Re-Implementing a Dynamic Field Theory Model of Mental Maps using Python and Nengo

Rabea Turon (rabea.turon@uniklinik-freiburg.de)[1]
Paulina Friemann (friemanp@cs.uni-freiburg.de)[1]
Terrence C. Stewart (terrence.stewart@nrc-cnrc.gc.ca)[2]
Marco Ragni (ragni@informatik.uni-freiburg.de)[1]
[1]Albert-Ludwigs-Universität Freiburg, Germany
[2]National Research Council of Canada, University of Waterloo Collaboration Centre, Canada

Abstract

In Dynamic Field Theory (DFT) cognition is modeled as the interaction of a complex dynamical system. The connection to the brain is established by smaller parts of this system, neural fields, that mimic the behavior of neuron populations. We reimplemented a spatial reasoning model from DFT in Python using the Nengo framework in order to provide a more flexible implementation, and to facilitate future research on a more general comparison between DFT and the Neural Engineering Framework (NEF). Our results show that it is possible to recreate the DFT spatial reasoning model using Nengo, since we were able to duplicate both the behavior of single neural fields and the whole model. However, there are statistical differences in performance between the two implementations, and future work is needed to determine the cause of these differences.

Keywords: Dynamic Field Theory; Nengo; mental maps; model reimplementation; spatial relational reasoning

Introduction

The way humans build maps from descriptions of relations of objects to each other is not yet fully understood. One approach to model how these maps arise is the spatial reasoning architecture from Kounatidou, Richter, and Schöner (2018). Their model implements an application of the Dynamic Field Theory (DFT), which views cognition as the development of a complex dynamical system (Schöner, Spencer, & Group, 2016). The model can be supplied with sentences that describe the relative location of two colored objects in 2-dimensional space, e.g., "There is a cyan object above a green object". From that, it builds a spatial scene in a 2-dimensional space, represented by activity in a 2-dimensional sheet of simulated neurons. A more complex spatial scene with more objects can be built by supplying multiple sentences. The implementation of the spatial reasoning architecture is realized in the Graphical User Interface (GUI) framework cedar (Lomp, Zibner, Richter, Ranó, & Schöner, 2013).

In this paper, we present a re-implementation of this model from Kounatidou et al. (2018). Rather than using their graphical framework, we implement the various components using Python, and then use the neural modelling toolkit Nengo (Bekolay et al., 2014) to combine the components together. There are two main reasons for this endeavor: The cedar framework, and therefore the spatial reasoning model itself, is difficult to modify and to use in scenarios that are different to the one presented in Kounatidou et al. (2018). Its application to future research as a general-purpose cognitive model of spatial reasoning is therefore limited by its implementation. The second reason to re-implement the model in Nengo is to approach a more general comparison between NEF and DFT using the spatial reasoning architecture as an example.

In the following, we will first briefly describe DFT and the spatial reasoning architecture. We will then specify the cedar model components used in this architecture, and describe the reimplementation in Nengo. Lastly, we compare the two implementations using an examplary spatial scene.

The spatial reasoning model

Background

The assumption of Dynamic Field Theory (DFT) is that cognition and behavior arise from the brain's development as a complex dynamical system (Schöner et al., 2016). Attractors in this system are then "functionally significant states of cognitive processes". An example for such an attractor of the spatial reasoning architecture is the spatial scene that results after supplying it with relational information.

The complex system in DFT consists of many subparts which DFT calls neural fields. They model the activation of populations of neurons. A complex system like the spatial reasoning architecture consists of multiple neural fields and some additional computations between these. Neural fields themselves are dynamical systems, too, whose dynamics are described by a differential equation:

$$\tau \dot{u}(x,t) = -u(x,t) + h + s(x,t) + \int k(x-x')g(u(x',t))dx' \quad (1)$$

In this equation $u(x,t)$ is the activation of the neural field at location x and time point t, h is the resting level of the neural field, $s(x,t)$ is some external input to the field, and integral computes local interactions in the field. The neural fields described by this equation can be of different dimensionality. What they have in common is that they can form peaks of activation that get passed on to other neural fields and can lead to further peaks there.

To make the building of complex dynamical systems as easy as possible DFT researchers have built software that helps with this task. In the case of the spatial reasoning architecture this software is cedar, a graphical-user-interface where computational elements can be added to an architecture with a simple drag and drop interface (Lomp et al., 2013). In addition to neural fields other elements can be added to the archi-

tecture, like inputs to the system or projections from a lower to a higher dimensional space.

Importantly, the spatial reasoning architecture developed in Kounatidou et al. (2018) was not created with the current version of cedar and does not perform correctly in the current version. The version that was used for this project can be found in an article by the Autonomous Robotics Group (2018). Models created in cedar can be saved to and loaded from JSON files.

The architecture

An overall image of the spatial reasoning architecture from Kounatidou et al. (2018) can be found in Figure 1. It consists of five conceptually distinct parts.

The first part of the architecture deals with the concepts that can be activated by the user. These are the spatial relations and the objects that are placed in a scene. The architecture is able to represent up to five different objects which are identified by their color, i.e. a red object, a blue object, a cyan object, a green object and an orange object. The translation from these colors to a continuous space is implemented by mapping them to the hue dimension. For all objects two input nodes exist since in each supplied sentence it has to be specified if an object is the reference object of the sentence or the target object. There are four spatial relations corresponding to the cardinal directions *Left*, *Right*, *Above* and *Below*.

The second part of the architecture is the attentional system. This system is responsible of 'attending to' or activating objects (i.e., the colored objects described earlier) that are already in the scene or that should be added to the scene in a new place, depending on the interaction with other neural fields from outside the attentional system. It consists of the color attention field and the 3-dimensional attention field and forms peaks for objects that are attended.

The scene representation forms the third part of the architecture. This includes the scene representation field where the locations of the objects are depicted over two-dimensional space while the third dimension of the field depicts the color of the existing objects.

The fourth part of the architecture is the spatial transformation and object creation system. It enables the architecture to place objects according to the relational premises that are supplied to the system. It represents each part of a premise, i.e. the reference object, the target object and the relation in a separate neural field and performs the transformations that are necessary to place a new object in the scene.

The fifth part of the architecture is responsible for the organization of all processes. It consists of intention nodes that determine whether a process is currently active or not and of condition-of-satisfaction (CoS) nodes that represent whether a process has been finished successfully. A more detailed description of the five parts can be found in the original paper.

Model components in cedar

A complex model in cedar is built from a set of basic components. The following components are needed for this model:

- `NeuralField`: This module implements the neural field equation from dynamic field theory. After an update step with the neural field equation a sigmoid function is applied to the activations before passing them on to another module.
- `GaussInput`: A GaussInput module is one of the input modules of cedar. This means that it does not receive any input but constantly sends the same signal. In the case of the GaussInput this signal is a two-dimensional gaussian peak whose peak position and amplitude are defined as parameters of the module.
- `ConstMatrix`: This module is another input module. It sends a constant 2-dimensional matrix output with one constant value at all positions of the matrix.
- `SpatialTemplate`: The SpatialTemplate is another input module. With the right parameters it creates a funnel-like pattern directed towards one of the cardinal directions *right*, *left*, *above* or *below*.
- `Projection`: The Projection module projects an input to a different dimensionality. It can either upscale an input from a lower dimension to a higher dimension by repeating values along the new dimension or it can downscale an input from a higher dimension to a lower dimension by performing a compression along the dimensions that should be reduced. As a compression operation the sum, maximum, minimum or average along a dimension can be used.
- `StaticGain`: The StaticGain module multiplies an input by a constant value that can be set as a parameter.
- `Boost`: The Boost module is another input module. It sends a scalar value that can be set as its strength parameter. However, it can be set to being active or not active during a simulation. Depending whether it is active or not it either sends no signal or the strength value defined. The Boost module is the module through which changing input is supplied to a system.
- `ComponentMultiply`: The ComponentMultiply module performs a componentwise multiplication of two inputs. If the two inputs have exactly the same shape this is an elementwise multiplication. Otherwise one of the inputs has to be of a lower dimensionality and each of its values is then multiplied with the other input's values along the additional dimension.
- `Convolution`: The Convolution module takes two inputs and performs a convolution on one of the inputs with the other input as the kernel.
- `Flip`: The Flip module receives a two-dimensional input and flips it along the first dimension, the second dimension or both.
- `Group`: A container to organize other components.

Nengo

Nengo (Bekolay et al., 2014) is a software tool that was originally created to build and simulate large-scale neural models based on the Neural Engineering Framework (NEF) (Eliasmith & Anderson, 2003). More recently, the toolkit has

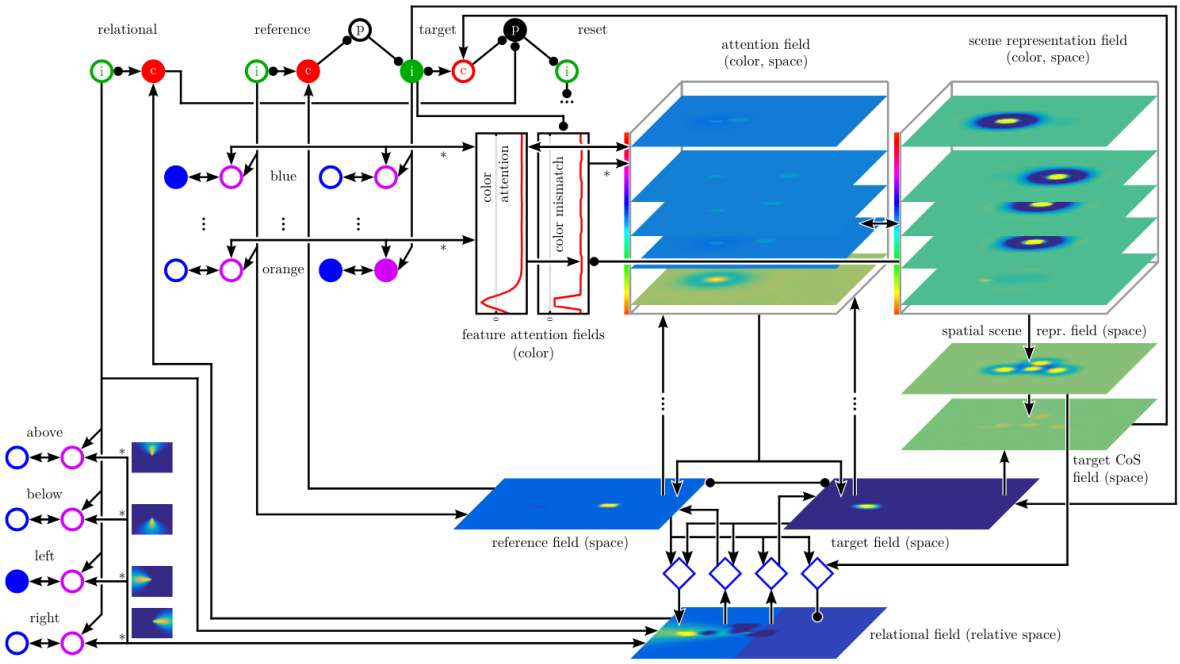


Figure 1: A conceptual image of the DFT spatial reasoning architecture from Kounatidou et al. (2018) reprinted from the original paper. The blue and pink circles signify nodes that represent the input concepts of the architecture. The attentional system consists of the feature attention fields in the middle and the attention field to the right of these. The scene representation fields can be seen on the right of the image. At the bottom the fourth part of the architecture, the *spatial transformation and object creation system* can be seen. The intention nodes are depicted as the red circles at the top, the CoS nodes as the green circles.

expanded to support deep learning and vector-based cognitive modelling, making Nengo now support a wider range of modelling approaches. Most importantly, Nengo provides a simple Python interface for defining new components, and then the standard Nengo framework can be used for combining these components, running simulations, and gathering data. By constructing our re-implementation in this way, we can to run any cedar model that uses the components we have re-implemented simply by taking the cedar version, saving it as a JSON file, and loading that saved JSON file into Nengo.

Model components in Nengo

The Nengo implementation of the cedar modules is based on the Nengo `Node` object. Some of the cedar modules (e.g. the `NeuralField`) require an update of their state with each simulation step while others (e.g. the `GaussInput`) pass on a constant signal during the whole simulation. The `Node` object provides easy-to-use functionality for both of these options. In the first case one can define an update function that depends on time and initialize a `Node` instance with this update function as the output parameter. In the latter case the constant signal is simply passed to a `Node` instance as the `output` parameter. In the implementation of the cedar modules in Nengo each cedar module has a corresponding object class of the same name. After initialization a Nengo `Node` instance can be created from this class's instance by calling the `make_node()` method. The `Node` instance is then accessible via the `node` attribute.

The Nengo implementation of the cedar modules is based on Schöner et al. (2016), as well as on the cedar documentation and the cedar source code (Autonomous Robotics Group, 2018). Another source of information is the behavior of the modules in cedar. For each module test instances of the cedar modules were created to observe their behavior for different parameter settings or different inputs. The observed behavior of the modules is also the principal validation for a correct implementation.

Since the goal was to implement the spatial reasoning architecture, not all of cedar's functionality had to be implemented. This means only the cedar modules that are part of the spatial reasoning architecture were implemented in Nengo. Moreover, some parameters of the cedar modules were not implemented in Nengo if they are not used in the spatial reasoning architecture or if their value is constant among all instances of the spatial reasoning architecture.

Nengo Results

Verification of NeuralField implementation

To make sure that the neural field equation is implemented correctly the temporal development of the `NeuralFields` of

cedar and Nengo were compared. To examine different parameters (e.g. different dimensionality or different border types) several `NeuralFields` with gaussian input were created and compared visually. For the visual comparison color maps of the `NeuralFields'` activation, their lateral interaction and their sigmoided output were created for both the cedar and the Nengo simulations. The appearance of the color maps and the mininum and maximum values were used as a measure of resemblance. To avoid random fluctuations the noise values were set to zero in these comparisons.

Apart from the identical evolution of the neural fields in terms of activation values the timing of this evolution was another tested aspect. To compare the timing the maximum activation value for the comparisons from above was tracked and contrasted in a plot. Moreover this comparison was performed for several different values of `tau`. Figure 2 contains one such comparison for three different tau values. As can be seen, each pair of curves with the same configuration progresses similarly through time, suggesting that the timing of the two implementations is the same.

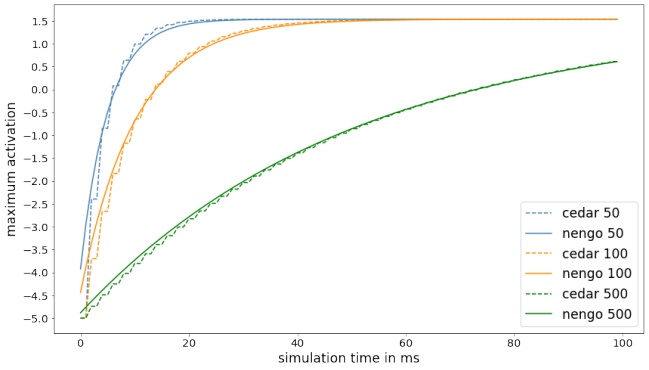


Figure 2: Maximum activation of a test NeuralField for different tau values in cedar and Nengo.

One Spatial Scene in Detail

Kounatidou et al. (2018) gives one example for a spatial scene that can be created with the spatial reasoning architecture. The scene is created from four successively supplied sentences:

1. There is a cyan object above a green object.
2. There is a red object to the left of the green object.
3. There is a blue object to the right of the red object.
4. There is an orange object to the right of the red object.

This scene was used as a test to see if the whole spatial reasoning architecture in Nengo works and to compare the scenes that arise in the original implementation in cedar and in the Nengo implementation. Figure 3a shows the development of the scene in cedar. The precise temporal sequence of inputs needed to create this scene is given in Table 1. The same experiment structure was used in the Nengo simulations.

In Figure 3b the development of the scene in Nengo can be seen. Even though the scenes of cedar and Nengo are slightly different, overall scene arrangement and development are the same. Differences in the development of the scene are also normal in separate runs in cedar due to additive random noise in the neural fields.

Simulation time

While our results show that the Nengo version of the model works, our initial Python implementation is much slower than the cedar version. When running in cedar, there is a "Factor for the fake DT" (default 0.26) which controls the time resolution of the simulation of the dynamic equations. In Nengo, the default time resolution is 1ms. This meant that a simulation which takes 2.3 minutes in cedar took 180 minutes in Nengo, i.e., a speed factor of .013.

While Nengo does allow components to define their own adaptive time step, we have not yet implemented this. Instead, we adjust the time step by a factor `tau`, where `tau`=1 is the original (1ms per time step) and tau=0.01 would be 100ms per time step. Note that this is the same as the `tau` parameter in the NeuralFields definition. If `tau` is decreased, this leads to an increase in the step size because the tau parameter is the divisor of every update step. This adaptation can not be performed up to any arbitrary factor since the update steps are a discretization of a continuous time process and at some point this discretization is too inaccurate to capture the original development. To determine a stable value for the factor of tau at least five simulation runs of the test scene were run for different tau factors. The rate with which a `tau` factor led to the scene predicted by the cedar model and the simulation times for different `tau` factors can be seen in Table 2. The test with the different `tau` factors also revealed that the standard update size of 1 does not reliably lead to the correct scene but seems to be rather unstable since it only lead to the correct scene in 2 out of 9 runs.

As can be seen in Table 2, simulations with a tau factor below 0.15 do not always lead to the correct scene representation. Some of these failed simulations are depicted in Figure 4. For these simulations it is likely that the update steps are too big and processes that would go in the opposite direction as the previous step can not be integrated due to the few updates. However, there are `tau` factors smaller 1 for which the scene seems to reliably develop correctly and which therefore can provide a time improvement.

We are still looking into other optimizations that we believe could help speed up the Python implementation of the DFT equations, which are most of the computation time in this model. For all subsequent experiments the `tau` factor of 0.15 was used, since it provided the fastest simulation times while maintaining high confidence to result in the correct scene.

Testing Results

To explore the behaviour of our re-implementation of the DFT mental map model, we generated a set of test inputs

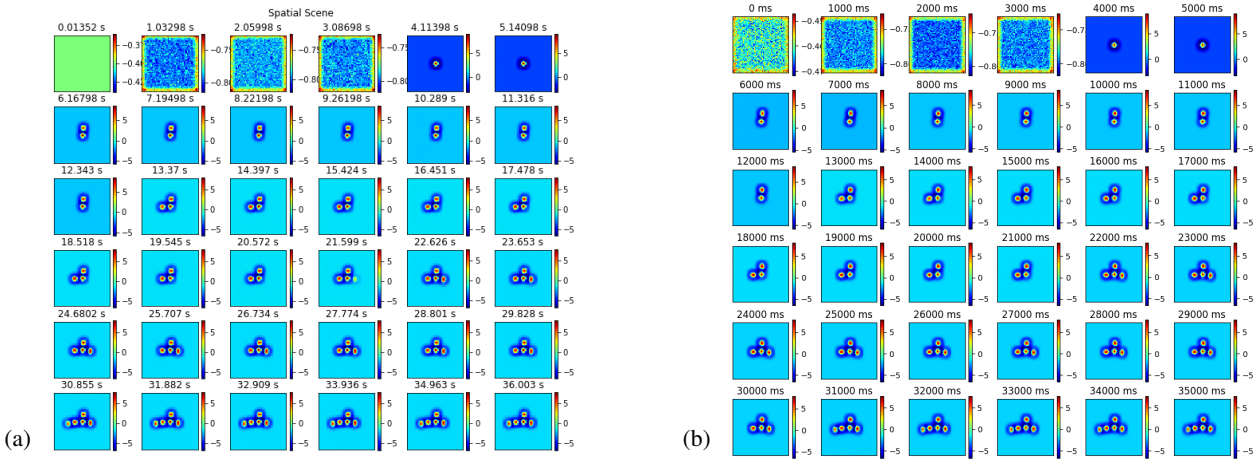


Figure 3: Evolution of the spatial example scene in cedar (a) and Nengo (with updates every ms) (b) with the default parameters.

Table 1: Instructions from the cedar experiment file.

simulation time	actions
0.0s	Activate the Boost modules ”Reference: Green”, ”Spatial relation:Above” and ”Target:Cyan”.
0.5s	Deactivate the Boost modules ”Reference: Green”, ”Spatial relation:Above” and ”Target:Cyan” and activate the Boost module ”Action:Imagine”.
9.0s	Activate the Boost modules ”Reference:Green”, ”Target:Red” and ”Spatial relation:Left”.
9.5s	Deactivate the Boost modules ”Reference:Green”, ”Target:Red” and ”Spatial relation:Left”.
18.0s	Activate the Boost modules ”Reference:Red”, ”Target:Blue” and ”Spatial relation:Right”.
18.5s	Deactivate the Boost modules ”Reference:Red”, ”Target:Blue” and ”Spatial relation:Right”.
27.0s	Activate the Boost modules ”Reference:Blue”, ”Target:Orange” and ”Spatial Relation:Left”.
27.5s	Dectivate the Boost modules ”Reference:Blue”, ”Target:Orange” and ”Spatial Relation:Left”.
36.0s	End the experiment.

Table 2: Simulation times and success rate of the test scene for different tau factors in Nengo.

tau factor	simulation time	success rate
0.02	3.5 minutes	0/7
0.05	9 minutes	3/10
0.1	17 minutes	9/11
0.15	26 minutes	5/5
0.2	35 minutes	8/8
0.5	1.5 hours	6/6
1.0	3 hours	2/9

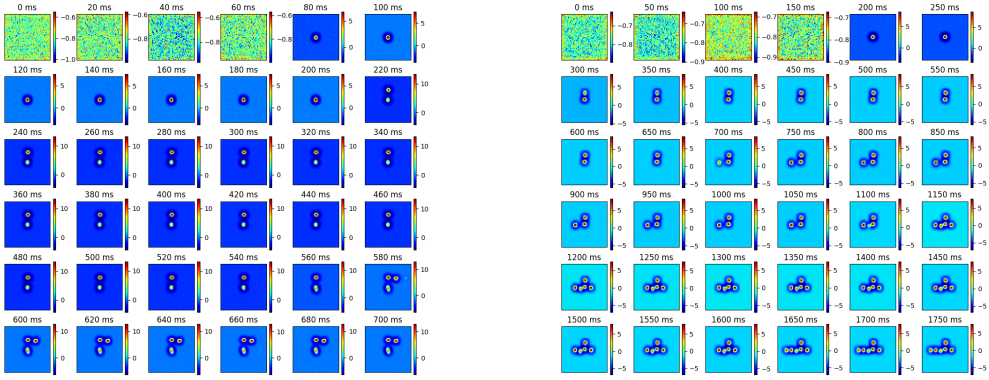
describing from two to four relational premises with all combinations of directions to generate a resulting scene. These inputs were run in both cedar and Nengo for comparison (see Figure 5 for examples).

Given these input scenes, we measured the proportion of time the models generated a correct final representation, i.e., the representation predicted by the cedar model. We knew that for some of the scenes the model would not create a scene consistent with the input statements due to the phrasing of the statements. However, they still gave us some information about the workings of the models. Since the models introduce random variability, we ran each input multiple times. Interestingly, the Nengo implementation was found to be more reliable than the cedar version; the Nengo simulations resulted in a correct scene in 73.26% of its simulations while the cedar simulations lead to the right scene in only 50.23% of its simulation runs. Determining the cause of this difference is the topic of ongoing work.

It should also be noted that for each test input, there was always at least one simulation run in cedar that resulted in the same mental map as a Nengo simulation run. This indicates that the models are doing similar things in those runs. For this reason, we believe that the core Python re-implementation in Nengo is working correctly, but that there are subtle differences with time steps and noise that are causing the differences in behaviour.

Conclusion and Future Work

Our goal was to reimplement the spatial reasoning architecture from Kounatidou et al. (2018) in the Python framework Nengo (Bekolay et al., 2014). The findings from the results section suggest that this goal is achieved, in that the system produces the desired behaviour. However, there are significant differences between the implementations that may be



(a) Tau factor 0.02

(b) Tau factor 0.05

Figure 4: Failed scene evolution of the example scene (see Table 1) in Nengo for different tau factors. The shown evolutions are only exemplary and can be different in each run.

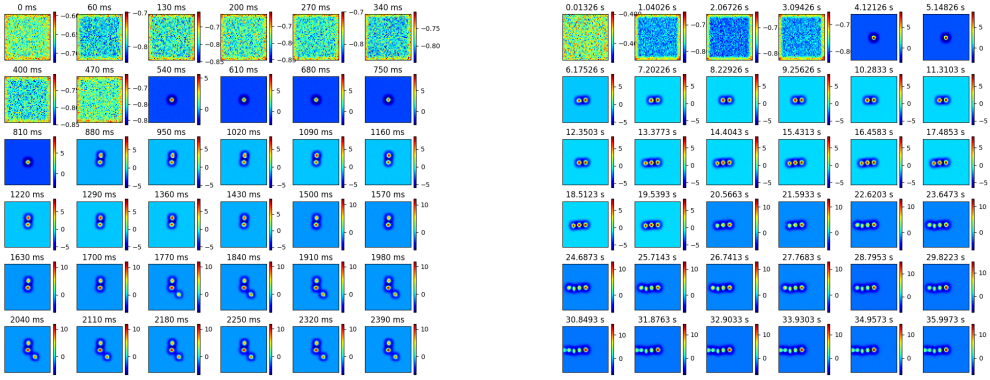


Figure 5: Example scenes from our test datasets. The scene development on the right is from a cedar simulation. The scene on the left is from a Nengo simulation.

due to the precise details of the random noise and the time steps used for calculation.

Importantly, since these low-level implementation details affect the overall performance of the model (as seen in the variability in the testing), understanding exactly what is causing these differences is important for interpretting any DFT model. We intend to continue to analyze these details and characterize them.

References

Autonomous Robotics Group, R.-U. B., Institut fuer Neuroinformatik. (2018). *Cedar 2018.* `https://github.com/cedar/cedar/tree/eeda0d6f79f5a0e420a877c642ce1b9ff48ba8dd`. GitHub.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., ... Eliasmith, C. (2014). Nengo: a python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, *7*, 48.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Kounatidou, P., Richter, M., & Schöner, G. (2018). A neural dynamic architecture that autonomously builds mental models. In *Cogsci.*

Lomp, O., Zibner, S. K. U., Richter, M., Ranó, I., & Schöner, G. (2013). A software framework for cognition, embodiment, dynamics, and autonomy in robotics: cedar. In *International conference on artificial neural networks* (pp. 475–482).

Schöner, G., Spencer, J. P., & Group, D. R. (2016). *Dynamic thinking: A primer on dynamic field theory*. Oxford University Press.

Integrated Model of Fatigue and C-17 Approach and Landing Operations

Bella Z. Veksler
(b.veksler@tier1performance.com)
Tier1 Performance Solutions
Covington, KY 41011 USA

Megan B. Morris
(megan.morris.3@us.af.mil)
Air Force Research Laboratory
WPAFB, OH 45433 USA

Michael A. Krusmark
(michael.krusmark.ctr@us.af.mil)
L3Harris Technologies
Melbourne, FL 32919 USA

Glenn Gunzelmann
(glenn.gunzelmann@us.af.mil)
Air Force Research Laboratory
WPAFB, OH 45433 USA

Abstract

Fatigue is a common occurrence in several occupational fields, often resulting in operator performance and health issues. Biomathematical models of fatigue have become useful tools in several fatigue risk management programs. However, these tools still have limitations in terms of identifying specific performance outcomes affected by fatigue, as well as individualizing fatigue estimates to individual operators. The integration of computational cognitive models and biomathematical models can help solve these issues in a complex operational context. The current effort aims to develop an integrated model of fatigue in the context of C-17 approach and landing operations. Specifically, we integrate a biomathematical fatigue model with a task network model to estimate performance degradation due to fatigue. The following paper outlines the development of the task network model and integration with the biomathematical fatigue model.

Keywords: task network model; biomathematical fatigue model; fatigue; aviation

Fatigue is a pervasive issue in work environments involving factors such as long work hours, shift schedules, circadian desynchrony, and high workload tempo, factors common in transportation, healthcare, and law enforcement, among other fields. Fatigue often results in performance degradations and can have significant negative effects on operator health, especially if fatigue is chronic in nature (Belenky, Lamp, Hemp, & Zaslona, 2014; Craig & Cooper, 1992). Biomathematical fatigue models are promising predictive tools in fatigue risk management (FRM) programs in high-risk operational settings. These models commonly use factors such as homeostatic regulation, sleep/wake schedules, and circadian rhythm to create general predictions of fatigue for operators (Mallis, Mejdal, Nguyen, & Dinges, 2004). However, these models have limitations that affect the accuracy of fatigue predictions. They tend to predict general performance outcomes (e.g., cognitive effectiveness) that might not relate directly to risk in an operational setting. Additionally, these models commonly lack individualization; rather, they give fatigue predictions for an "average" operator (Civil Aviation Safety Authority, 2014; Dawson, Darwent, & Roach, 2016; Mallis et al., 2004). Computational cognitive models can provide a cost-effective and flexible means to explore the usability of systems through simulation (Pew, 2007). Recently, research has successfully integrated biomathematical models of fatigue with cognitive architectures (e.g., Gunzelmann, Veksler, Walsh, & Gluck, 2015). In the current effort, we work toward developing an integrated model that can pinpoint specific performance degradations due to fatigue in a complex real-world environment, and allows the inclusion of individual difference modulations. Specifically, we integrate a biomathematical fatigue model with a task network model (Laughery, Archer, Plott, & Dahn, 2000) to predict C-17 aircraft approach and landing performance degradations.

Background

C-17 mobility pilots and aircrew are especially susceptible to fatigue given unique characteristics of the operational environment. Basic crews and augmented crews have flight duty periods of up to 16 and 24 hours, respectively. Flight legs commonly cross multiple time zones during missions and missions comprising multiple legs often last several days. Research suggests mobility aircrew are commonly fatigued during missions and believe that changes need to be made in the mobility community to address fatigue (Morris, Howland, Amaddio, & Gunzelmann, 2020; Morris, Veksler et al., 2020). Currently, the United States Air Force Air Mobility Command (AMC) uses the Fatigue Avoidance Scheduling Tool (FAST®; Hursh, Balkin, Miller, & Eddy, 2004) and underpinning biomathematical fatigue model, Sleep, Activity, Fatigue, and Task Effectiveness (SAFTE™; Hursh, Redmond, et al., 2004) to develop mission effectiveness graphs that balance fatigue with operational needs and recommend sleep schedules for aircrew based on mission information (e.g., flight leg start and end times, time zones, light). The SAFTE model includes a circadian process affecting sleep regulation and performance. Within the model there is a reservoir capacity which refers to an

individual's maximal capacity to perform tasks. This capacity is affected by sleep and wakefulness. When an individual is awake, the reservoir level decreases, and when an individual is asleep the reservoir is replenished. Accumulation in the reservoir is affected by sleep intensity and sleep quality. Sleep intensity is a function of time-of-day (circadian phase) and the current reservoir level (sleep debt). Performance is affected by the sleep reservoir, circadian phase, and sleep inertia. Performance output from the model is a general cognitive effectiveness in the form of a percentage from 0 to 100% (Hursh, Redmond, et al., 2004). Aircrew can use the resulting mission effectiveness graph from FAST and SAFTE to plan fatigue mitigation strategies. It is not known which performance metrics are affected in C-17 pilot operation, nor is there currently a mechanism to individualize these predictions within AMC's general risk management program. As a result, additional tools are needed to provide insights into specific performance metrics that are likely to be affected by fatigue and have the ability to incorporate individual difference factors that affect fatigue.

The current C-17 approach and landing model was developed using a task network model. These models are comprised of nodes and connections that let activity flow through the network and provide an efficient way of simulating the complexities of operator/system designs (Hansberger & Barnette, 2005; Schunk, 2000; Swoboda, Katz, & Kilduff, 2005). The sequencing of various tasks and subtasks is integrated into task network models and each task/subtask is assigned to an operator. Operators, in turn, have properties specific to their role in the simulation and individual differences can be incorporated in the operator profile by specifying amount of military training, length of service, age, rank, time in position, and workload threshold (Hansberger & Barnette, 2005; Richardson, Mittrick, & Hanratty, 2016; Swoboda et al., 2005). Tasks within the model have preconditions that must be met to execute the task. When a task is executed, the state of both the environment and the operator change. Due to the design of the task network, certain subtasks can be completed concurrently (especially if multiple operators are present) as long as the operator's workload threshold has not been exceeded. Task network models have been used in many military and commercial applications to make predictions about performance under varying conditions (e.g., Bloechle & Schunk, 2003; Laughery et al., 2000; Schunk, 2000).

Our task network model was developed in the C3TRACE (Command, Control and Communication Techniques for Reliable Assessment of Concept Execution; Kilduff, Swoboda, & Barnette, 2005) task-network modeling environment to represent the relevant tasks and subtasks involved in approach and landing phases of flight for the C-17 aircraft. C3TRACE is a modeling environment in which tasks, transitions, and operators can all be represented in a network model. It is owned by the US Army Research Lab (ARL-HRED) and was developed by Micro Analysis & Design (acquired by Alion Science and Technology) (Plott, 2017). C3TRACE allows the modeler to encode the relevant task information flow and then uses a stochastic discrete event simulator (the simulation engine is Micro Saint Sharp (Bloechle & Schunk, 2003)) to output results of the simulation. The modeler can then evaluate various aspects of task performance such as operator workload, task execution time, etc. In particular, C3TRACE allows the modeler to define the magnitude of the workload components for Visual, Auditory, Cognitive, and Psychomotor (VACP) aspects of performance for each task/subtask. Tasks can also contain logic regarding execution time, "if/then" rules to dictate when tasks can be "released", and the capacity to modify environmental variables as needed to simulate task effects (i.e., adjustments to plane position). In the past, C3TRACE has been utilized in simulating high-level team interaction because of its ability to integrate multiple personnel and personnel groupings into a model that selects operators based on availability (workload-based) and task priority. Furthermore, personnel characteristics can be modified to better reflect operator experience (i.e., education level, age, rank, time in position, workload threshold, etc.) that in turn can influence task performance (Cosenzo, Kilduff, & Swoboda, 2005).

Model Development

Approach and landing tasks and pilot and co-pilot interactions were developed based on an existing analysis of standard procedures and through discussions with two experienced C-17 pilots. The model was divided into two tasks: approach and landing as defined in the procedures. Each task was composed of several subtasks that had to be performed in a certain order, although some subtasks could be done concurrently as they required either the pilot or co-pilot to perform them (see Figure 1).

Approach and landing both have strong monitoring components (see Figure 2) as the pilot and co-pilot must maintain basic airplane operations such as keeping the plane level and slowly descending in altitude as the plane approaches the runway while simultaneously performing the necessary subtasks to ensure a safe landing (i.e., setting and checking altimeters, doing approach and landing checklists, setting the flaps, verifying glideslope, lowering the landing gear, etc.). The task analysis also indicated another monitoring task that could potentially alter the plane's course if a threat was detected and a corresponding set of subtasks needed to be performed if that occurred. All monitoring subtasks were implemented in the model as concurrently occurring during the main approach and landing subtasks. Therefore, the operator's attention had to be switched between the main task and the monitoring components.

Several environmental variables were included in the model to simulate the plane flying and descending. Those environmental variables were controlled by a "dummy" operator that continuously updates the plane's state variables both in response to the pilot/co-pilot interaction

and by a basic linear drift model (see Figure 2). As the model runs, the plane gradually descends to the runway and the speed of descent (both X and Y direction) is modulated by variables such as flap settings.

The task analysis informed the model's timing of each subtask. C3TRACE allows the modeler to specify the mean and standard deviation of all timing intervals. For simplicity, the mean for each procedure was set to an estimate derived from the expert task analysis combined with input from the experienced pilots. A standard deviation of 10% was introduced to these estimates to add some stochasticity to the model's results. This value was used as a stand in and may change with future work; however, it does not affect the comparisons of the models discussed below. As per the task analysis, each subtask was assigned an operator or operator group (if either the pilot or co-pilot could perform the task). The task analysis also provided us with a breakdown of which VACP components were utilized in each subtask and those were set accordingly to simulate the workload associated with each subtask. Certain subtasks also altered state variables (i.e., flaps).

C3TRACE allows the starting conditions to be modified for each model run (i.e., the starting altitude of the plane and its descent rate) which can be utilized to produce predictions about when certain milestones will be reached during approach and landing (i.e., flaps are set, runway reached, checklists accomplished). These predictions can then be compared to data collected from real C-17 landings to verify the model's validity.

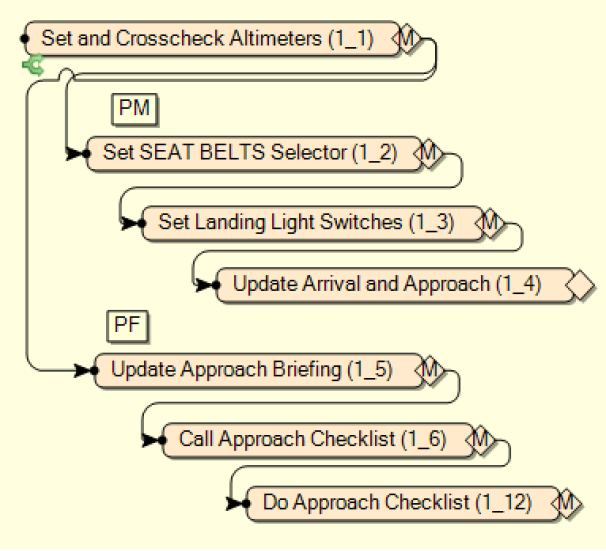


Figure 1: A sample set of subtasks required for approach. PM and PF indicate which operator typically does those tasks in order (PM: Pilot Monitoring, PF: Pilot Flying).

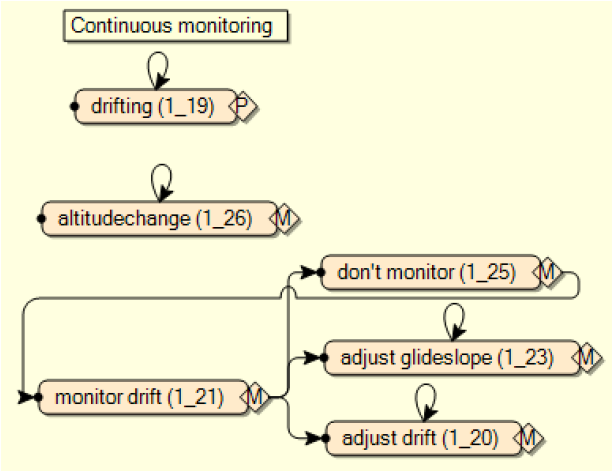


Figure 2: Snapshot of one monitoring component of the model. The *drifting* and *altitudechange* tasks are continuously performed by a "dummy" operator that updates the plane's position. The monitoring loop below involves the pilot or copilot making adjustments to the plane as needed to keep it on course.

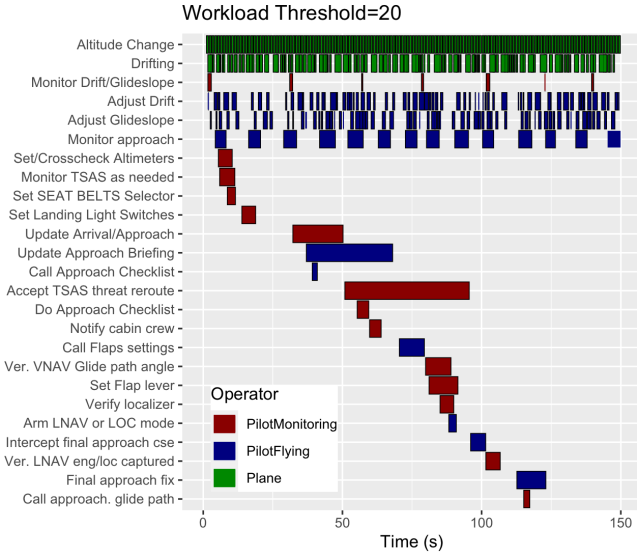
Fatigue Modulation

As an initial approach to modifying performance on this task and to simulate fatigue, we utilized the workload threshold parameter that can be set for each operator. The scheduler in C3TRACE assigns tasks to operators as long as (1) all preconditions for a given task are met and (2) the operator's workload threshold is not exceeded by the currently running tasks. With the inclusion of the monitoring tasks in this model, it is very possible for the pilot or co-pilot to be in the process of monitoring some state variable and adjusting it while attempting to perform the necessary approach and landing subtasks. In instances where the VACP workload is high for a given subtask (i.e., visually inspecting a dial while reporting the reading and adjusting something else), the monitoring component may pose some interference especially if the operator's current workload threshold is lowered (i.e., due to fatigue).

The effects of fatigue manifest themselves in this model by (1) reducing how many simultaneous tasks can be accomplished, (2) when those tasks are scheduled, and (3) how long those tasks will take to complete. This has further implications for important state variables such as deviation from the ideal glideslope as monitoring tasks may be delayed by other tasks. Figure 3 depicts the task timeline showing the various subtasks as they occur during model execution during the first 150s of approach and landing (shortened to fit within paper margins) under two settings of workload threshold for the operators (both operators' workload threshold is set to the same amount, either 20 or 8). Of note is that as the workload threshold is reduced, the frequency with which monitoring and adjusting takes place diminishes (see Adjust Drift and Adjust Glideslope tasks listed in Figure 3). Furthermore, other subtasks are more

staggered in their execution, prolonging the time to complete the required steps, as shown in the bottom graph where the bottom four subtasks are not even scheduled before 150s into the approach and landing procedure. Note that the number of simultaneous tasks that can execute is a function of both the workload threshold and the specific VACP components required for each task, so an exact number of simultaneous tasks will vary throughout model execution, but lower workload threshold will necessarily reduce the number of simultaneous tasks that the model can execute. The modification of workload threshold for operators is a good first approximation for modeling the deleterious effects fatigue has on performance.

We can inform the setting of the workload threshold by using fatigue estimates from biomathematical models of fatigue, in this case from the SAFTE model, and scaling the workload accordingly. Work is ongoing to determine the best way to do the scaling so as to produce changes in performance commensurate to those seen in human data. The output from the SAFTE model typically produces a performance effectiveness score on a scale of 0-100% by using sleep history (Hursh, et. al., 2004). In operational settings, performance effectiveness values higher than 77.5% indicate an alert individual, values between 70 and 77.5% indicate a moderately fatigued individual, and values below 70% indicate high fatigue and serious risk in continuing operating. In the approach and landing model described, a workload threshold setting of 20 would correspond to a relatively rested individual, whereas a setting of 8 would correspond to serious degradations in performance.



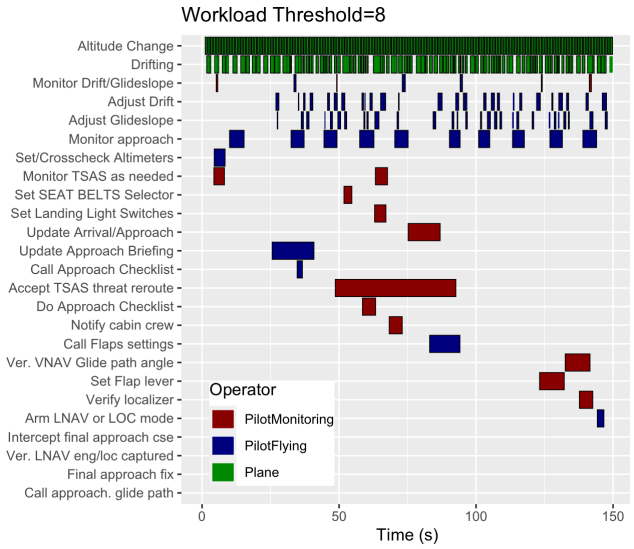


Figure 3: Task Timeline with different settings of operator workload threshold.

Model Results

The model was run 100 times using several settings of Workload Threshold for the two pilots. Preliminary results indicate differences in when important components of approach and landing procedures get executed (i.e., flaps deployed, speed brake set, gear lowered). In particular, lower levels of workload threshold resulted in significant delays and more variability in the timing of these subtasks (see Figure 4). In addition, there was an increase in the amount of drift observed throughout the model run as the operators made less adjustments to the plane (recall Figure 3's adjustment subtasks which are much more sparse in the WT=8 case).

There are many other diagnostic variables that we can observe in the output from a model run in C3TRACE which can be compared to real world landing data. Future work will integrate more of these variables.

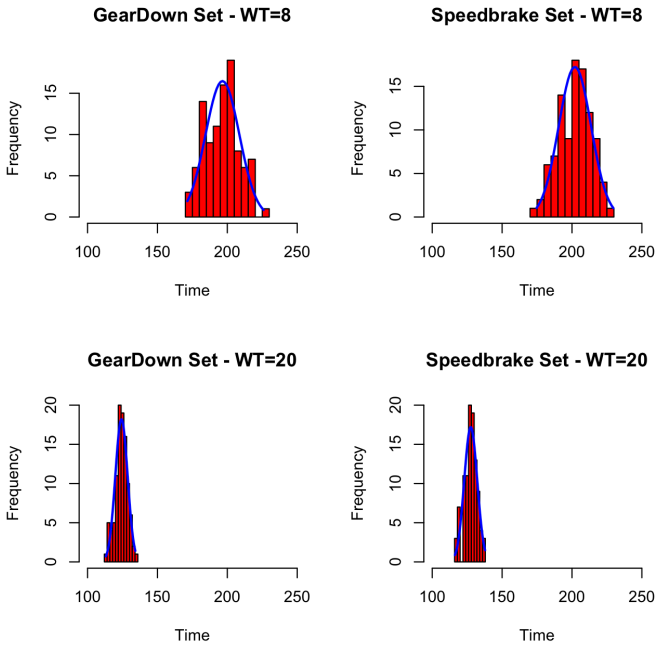


Figure 4: Distribution in timing of when gear is set down and when speed brake is set for two settings of workload threshold (100 model runs).

Discussion

We have developed an initial model integrating fatigue and a task network model of C-17 approach and landing operations. Currently, we modulate workload thresholds within the task network model with individualized fatigue estimates from the SAFTE model. SAFTE model estimates are derived from sleep estimates for individual operators. As a follow-on effort, we plan to validate the integrated model by fitting the predictions to performance metrics from actual C-17 flight data from an operational study. Sleep estimates will be derived from actigraph watches worn by pilots and co-pilots to generate fatigue estimates through SAFTE. The integrated model will allow us to identify specific performance degradations in the C-17 environment. This information can be used to develop more effective FRM programs and systems that link fatigue estimates to actual safety outcomes, a feature that is currently lacking in most FRM implementations (Dawson et al., 2016; Gander et al., 2017).

Acknowledgments

The opinions expressed herein are solely those of the authors and do not necessarily represent the opinions of the United States Government, the U.S. Department of Defense, the U.S. Air Force, or any of their subsidiaries, or employees. This research was partially supported by funding from the Defense Health Agency's Joint Program Committee on Military Operational Medicine. Distribution A. Approved for public release. Case number 88ABW-2020-2179.

.

References

Belenky, G., Lamp, A., Hemp, A., & Zaslona, J. L. (2014). Fatigue in the workplace. In M. T. Bianchi (Ed.). *Sleep Deprivation and Disease: Effects on the Body, Brain and Behavior* (pp. 243-268). New York, NY: Springer.

Bloechle, W. K., & Schunk, D. W. (2003). Micro Saint Sharp Simulation Software. *Proceedings of the 2003 Winter Simulation Conference*, 182-187.

Civil Aviation Safety Authority. (2014). Biomathematical fatigue models guidance document. https://www.iata.org/contentassets/5f976bb3ca2446f3a40e88b18dd61fbb/condensed-version-of-casa-biomathematical-models-doc.pdf

Cosenzo, K., Kilduff, P., & Swoboda, J. (2005). Human Performance Moderators for the C3TRACE Modeling Environment. Proceedings of the 14th Annual Conference on Behavior Representation in Modeling & Simulation. Los Angeles, CA: BRIMS Society, 2005.

Craig, A., & Cooper, R. E. (1992). Symptoms of acute and chronic fatigue. In A. P. Smith & D. A. Jones (Eds.), *Handbook of human performance* (pp. 289-340). San Diego, CA: Academic Press.

Dawson, D., Darwent, D., & Roach, G. D. (2016). How should a bio-mathematical model be used within a fatigue risk management system to determine whether or not a working time arrangement is safe? *Accident Analysis and Prevention*, *99*, 469-473.

Gander, P. H., Wu, L. J., van den Berg, M., Lamp, A., Hoeg, L., & Belenky, G. (2017). Fatigue risk management systems. In M. H. Kryger, T. Roth, & W. C. Dement (Eds.), *Principles and practice of sleep medicine* (pp. 697-707). Philadelphia, PA: Elsevier.

Gunzelmann, G., Veksler, B. Z., Walsh, M. M., & Gluck, K. A. (2015). Understanding and predicting the cognitive effects of sleep loss through simulation. *Translational Issues in Psychological Science*, *1*(1), 106-115.

Hansberger, J. T., & Barnette, D. (2005). Human performance modeling for operational command, control and communication. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *49*(12), 1182–1185.

Hursh, S. R., Balkin, T. J., Miller, J. C., Eddy, D. R. (2004). The fatigue avoidance scheduling tool: Modeling to minimize the effects of fatigue on cognitive performance. *SAE Transactions Journal of Aerospace*, *113*(1), 111-119.

Hursh, S. R., Redmond, D. P., Johnson, M. L., Thorne, D. R., Belenky, G., Balkin, T. J., … Eddy, D. R. (2004). Fatigue models for applied research in warfighting. *Aviation, Space, and Environmental Medicine*, *75*(3), A44-A53.

Kilduff, P., Swoboda, J., & Barnette, D. (2005). Command, Control, and Communications: Techniques for the Reliable Assessment of Concept Execution (C3TRACE) modeling environment: The tool. Technical Report: ARL-MR-0617.

Aberdeen Proving Ground, MD: U.S. Army Research Laboratory, HRED. https://apps.dtic.mil/dtic/tr/fulltext/u2/a435035.pdf.

Laughery, R., Archer, S., Plott, B., & Dahn, D. (2000). Task network modeling and the Micro Saint family of tools. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *44*(6), 721–724.

Mallis, M. M., Mejdal, S., Nguyen, T. T., & Dinges, D. F. (2004). Summary of the key features of seven biomathematical models of human fatigue and performance. *Aviation, Space, and Environmental Medicine*, *75*(3), A4-A14.

Morris, M. B., Howland, J. P., Amaddio, K. M., & Gunzelmann, G. (2020). Aircrew fatigue perceptions, fatigue mitigation strategies, and circadian typology. *Aerospace Medicine and Human Performance*, *91*(4), 363-368.

Morris, M. B., Veksler, B., Gaines, A., Krusmark, M., Jantscher, H., & Gunzelmann, G. (2020). Aircrew sleep schedules: A comparison of actual and prescriptive schedules. *To be presented at the 2020 Aerospace Medical Association Annual Scientific Meeting*, Atlanta, GA.

Pew, R. W. (2007). Some history of human performance modeling. In W. D. Gray (Ed.), *Integrated models of cognitive systems* (pp. 29-44). New York, NY: Oxford University Press.

Plott, B. (2017). Software user's manual for C3TRACE Command, Control and Communication: Techniques for the Reliable Assessment of Concept Execution (Version 3.8.0.8). ARL-HRED & Alion Science and Technology.

Richardson, J. T., Mittrick, M. R., & Hanratty, T. P. (2016). Modeling the impact of value of information on situational awareness using C3TRACE. Report No. ARL-TR-7684. Aberdeen Proving Ground, MD: US Army Research Laboratory. https://apps/dtic.mil/dtic/tr/fulltext/u2/1009847.pdf.

Schunk, D. (2000). Modeling with the micro saint simulation package. *Proceedings of the 2000 Winter Simulation Conference*, 274-279.

Swoboda, J. C., Katz, J. P., & Kilduff, P. W. (2005). A platoon level model of communication flow and the effects on soldier performance. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *49*(12), 1210-1214.

What Everyday Activities Reveal About Spatial Representation and Planning Depth

Petra Wenzl (pwenzl@uni-bremen.de) ✉
Institute for Artificial Intelligence
University of Bremen, Bremen, Germany

Holger Schultheis (schulth@uni-bremen.de)
Institute for Artificial Intelligence, Bremen Spatial Cognition Center
University of Bremen, Bremen, Germany

Abstract

Successfully performing everyday activities such as loading the dishwasher or setting the table relies on the involvement of many cognitive abilities. As such, everyday activities provide a unique window for investigating the involved cognitive abilities as well as their interaction, promising high ecological validity of the obtained findings. Against this background we investigated two cognitive abilities and their combination, which are crucial for virtually all everyday activities. Specifically, we investigated the nature of mental spatial representation and planning depth in rational planning by analyzing table setting behavior across many environments and actors. As recent modeling work indicates that rational planning is influenced by spatial properties of the environment, we investigate how representation of and reasoning about the spatial environment impact sequential action planning. Using a modeling approach, we compare models implementing different plannings depths and differently complex spatial representations. Our findings indicate that people plan opportunistically (one step ahead) and rely on a two-dimensional representation of their environment. These findings lend credit to the idea that humans minimize their cognitive effort (simpler representations, shallow planning) to efficiently perform everyday tasks.

Keywords: spatial cognition; rational planning; action sequences

Introduction

Everyday activities, such as cooking, cleaning, or setting a table, seem simple, but are in fact highly complex tasks involving many different cognitive abilities. Setting the table, for example, requires action and motor planning, navigation, spatial memory, action and motor control, and error monitoring and correction, among others.

We argue that everyday activities provide a unique and instrumental window for investigating the involved cognitive abilities. For one, everyday activities constitute *complex tasks* in the sense of Newell (1973) such that their study promises not only a deeper understanding of each of the abilities, but also of their interaction and integration. Furthermore, findings obtained from investigating everyday activities arguably offer higher ecological validity than findings obtained from experimental tasks commonly employed in the Cognitive Sciences. At the same time, everyday activities are still simple enough to also be investigated in the lab. Last but not least, understanding cognitive abilities in everyday activities is of great applied relevance by potentially allowing to better support people to live independently (e.g., in old age) without requiring professional aid.

Against this background, in this contribution, we investigate the nature of mental spatial representation and planning depth in rational planning by analyzing everyday activities. Planning and control of action sequences are necessary requirements for successful task performance in everyday life. In existing models of sequential action control (e.g. Botvinick & Plaut, 2004; Cooper & Shallice, 2006), the assumption seems to be that the to be controlled sequence is completely known from the outset, whereas we propose a stepwise approach. Recent modeling work suggests that rational planning is influenced by spatial properties of the environment, taking distance, relational dependencies (strong spatial cognition), and topology (containment) into account (Wenzl & Schultheis, 2020). Building on this modeling work, we examine how many dimensions people take into account when representing and reasoning about their spatial environment as well as how many steps ahead they plan their actions (i.e., planning depth). Our investigations take the form of a model comparison study, in which we develop and compare models realizing different dimensionalities of spatial representations and planning depths. The models are compared across three datasets of human table setting activities comprising various actors and environments. Modeling results indicate that people plan opportunistically employing a single-step look-ahead and that they rely on a two-dimensional representation of the environment, largely ignoring the vertical dimension.

The remainder of this paper is structured as follows: First, we give an overview of the role of rational planning, space, and minimization of cognitive and physical effort in the context of everyday activities. Subsequently, we investigate the role of planning depth and dimensionality of spatial representation using a modeling approach. We conclude with a discussion of our results and issues for future research.

Rational Planning, Space, and Minimization of Effort

Rational Planning

Mechanisms such as knowledge representation and cognitive processes have to be taken into account when trying to explain human behavior through rational analysis (Jones & Love, 2011). This is the core assumption of *bounded rationality* (Simon, 1955) which takes limitations in knowledge and processing capacity into account. To identify ef-

fective mechanisms that can plausibly be implemented by a resource-bounded human brain, computational modeling has been shown to offer a useful analysis tool (Icard, 2018).

Adaptive rationality proposes that good prediction methods are adapted to the structure of a given *local* environment, providing highly efficient solutions for a specific task (Schurz & Thorn, 2016). Human cognition generally is assumed to be locally optimal. This is consistent with research on sequential information search and planning which indicates that humans tend to use heuristic stepwise-optimal strategies rather than planning ahead (Meder, Nelson, Jones, & Ruggeri, 2019).

Taking the limitations of the human mind and the complexity of everyday activities into account, we propose that humans deal with such activities by using a rational planning strategy to choose their next action.

Space

All human activity takes place in space: Required items for a given (everyday) activity are located in the physical environment, and movement within this environment is necessary to perform the activity. Spatial properties, e.g., distance, are directly related to the required physical effort. While choosing the action sequence for performing a specific activity, the spatial properties of the environment may impose constraints, such as having to move one object first before being able to reach another object located behind it. Even if there are no hard constraints there are a number of reasons to believe that the order of action sequences is influenced by the spatial environment and its mental representation.

First, the organization of objects in physical space aims to minimize cognitive effort and to facilitate the performance of everyday activities (Kirsh, 1995). People use spatial arrangements to serve as cues what to do next by simplifying internal computation, e.g., by arranging objects in the kitchen in a way that it is obvious which vegetables need to be cut, washed, etc. in the next step. Minimizing computational effort by using the properties of the spatial environment to facilitate one's actions is also consistent with behavioral strategies relying on strong spatial cognition (van de Ven, Fukuda, Schultheis, Freksa, & Barkowsky, 2018) and cognitive offloading (Clark, 1996; Wilson, 2002) (see Minimization of Effort). Second, previous research has shown that the nature of mental representations of space has a marked influence on peoples behavior. Three-dimensional spaces seem to be represented in a "bicoded" way, splitting the representation in a metric planar representation of the plane of locomotion and a separate, possibly non-metric representation of the orthogonal space (Jeffery, Jovalekic, Verriotis, & Hayman, 2013). Human spatial navigation performance is significantly worse when navigating in a vertical environment than in a horizontal environment (Zwergal et al., 2016) and distance is represented with higher accuracy along the horizontal than the vertical axis (Hinterecker et al., 2018).

Taking the above considerations into account, we assume spatial properties of the task environment, i.e., distance, functional dependencies, and topology to be important factors when deciding for the next action.

Minimization of Effort

Hull's *"law of less work"* (Hull, 1943) states that physical effort tends to be avoided. Newer research indicates that physical and mental effort are equally aversive (Kool, McGuire, Rosen, & Botvinick, 2010). The concept of an internal cost of cognitive effort allows to explain the (globally) suboptimal strategies frequently observed in humans, as favoring simplifying strategies (heuristics) can be subjectively optimal when reducing the internal cost of mental effort outweighs the benefit of a more accurate strategy.

External scaffolding is a possible strategy to reduce cognitive effort (Clark, 1996). Accordingly, external structures are used to facilitate human problem-solving and to reduce the cognitive effort of a specific task by offloading (part of) the problem solution to external scaffolds such as tools or memory aids. Strategies to offload cognition are used particularly often in the context of spatial tasks (Wilson, 2002) (see Space).

Against this background, we assume that humans prefer planning strategies that locally minimize the effort required for task success.

Rational Planning Model for Table Setting

Consistent with the spatial environment being used to facilitate task performance, i.e., intelligent use of space (Kirsh, 1995), external scaffolding (Clark, 1996; Wilson, 2002), strong spatial cognition (van de Ven et al., 2018), and mental representation of space (Hinterecker et al., 2018), we expect specific spatial constraints to be of importance for planning.

Based on previous research evidencing that humans favor stepwise-optimal strategies over planning ahead (Meder et al., 2019) and the "law of less work" (Hull, 1943; Kool et al., 2010), we assume that the control of routine sequential actions, such as table setting, follows a strategy of rational planning. Taking the role of spatial properties in everyday activities into account, we propose that humans prefer specific action orderings: The next item to be picked up and taken to the table is assumed to be chosen based on the current location as well as the perceived cost of each possible action, with the lowest-cost action being chosen.

Employing a modeling approach, we examine the influence of the following spatial aspects of the task environment on action organization during table setting:

- *Distance*: minimizing traversed distance,
- *relational dependencies*: e.g., saucer goes below cup and should therefore be taken first, so both items have to be moved to and placed on the table only once, and
- *topology (containment)*: picking up items from, e.g., a counter top, is considered less effortful than picking up items stored in a closed cupboard.

We implemented our core assumptions in a computational model. The model approximates rational planning by determining the lowest-cost next action for each step from episode

start (no items on the table, subject at starting position) to task success (all required items on the table and – if specified – in the target position, subject standing in front of the table).

Each cost $C_{p,q}$ is calculated by determining the Euclidean distance between two item locations $p(x_1, y_1, z_1)$ and $q(x_2, y_2, z_2)$ in a nD representation of the specific environment, where n is either 1, 2, or 3. This distance is further qualified by relational dependencies (parameter k) and containment (parameter c) yielding a weighted cost computed as given in Eq. 1, where d is the Euclidean distance. Setting parameter k to a value < 1.0 decreases the weighted cost, thus corresponding to a higher probability of taking the item in question first, whereas setting parameter c to a value > 1.0 increases the weighted cost.

$$C_{p,q} = d(p,q)^k \cdot c \tag{1}$$

Relational dependencies are defined as constraints that favor putting one item on the table earlier than a second item, e.g., because the first item is supposed to be placed below the second item (saucer and cup, etc.) or because the item is used to define the place setting on the table (placemat, plate). Containment indicates whether an item can be accessed directly or whether it is stored in a cupboard or the like which has to be opened first.

We assume relational dependencies to have an influence on the ordering of items as, with an ideal ordering, each item has to be picked up and placed on the table only once, and the placement of subsequent items is facilitated (e.g., not having to know how much space to leave between items of silverware for the plate). In contrast to choosing an arbitrary ordering, in which items already on the table might have to be moved again (e.g., lifting the cup to place the saucer below it, or making space for the plate by moving the silverware), this ideal sequence minimizes the cognitive and physical effort. Since the opening of cupboards involves physical effort, containment is considered to be another cost factor. The weighted cost for each possible item also depends on which dimensions are considered when calculating the cost: Distances differ depending on whether they are computed in 1D (i.e., with respect to the x, y, or z axis), 2D (i.e., with respect to the xy, xz, or yz axes), or 3D (i.e., with respect to the xyz axes). Parameters k and c are treated as free parameters of the model and will be estimated from the data.

Simulations

Simulations aim to test two specific aspects: Planning depth and dimensionality. For this purpose, we conducted two model comparison studies: The purpose of the first simulation was to compare different levels of planning depth: whether the model assumes a one- or a two-step look-ahead (see Planning Depth). The second simulation examined the dimensionality of the spatial representation people employed for distance calculations (see Dimensionality).

Based on a given spatial layout with item coordinates, the task description (required items), and a sequence of current locations, simulations were conducted as follows: For each predicted next item, the prior location was taken as the current location, regardless of whether the corresponding action was a table setting action. In each step the cost for all next possible actions was calculated (Eq. 1, p = current location, q = item location), from which the item with the lowest associated cost was chosen to be picked up next (Fig. 1). If there were multiple items with the same associated cost, one item was chosen randomly.

Table 1: Parameter estimates for different items

Category of relational dependencies (k)	Items
strong	tray, placemat, table cloth
medium	plate (empty), napkin
none ($k = 1$)	all other items

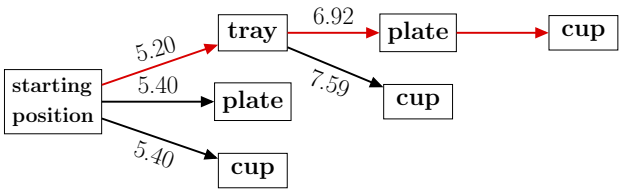


Figure 1: Example for stepwise-optimal item selection based on weighted cost (TUM environment, k and c set)

Parameters k and c were estimated by grid search. Parameter k was estimated per item category (see Tab. 1), i.e., items with strong relational dependencies (e.g., placemat), items with medium-strength relational dependencies (e.g., plate) and items without relational dependencies ($k = 1$). c was estimated for all objects in closed containers (e.g., cupboard, drawer). To evaluate how well the sequences generated by the model and the observed sequences matched, we computed the Damerau-Levenshtein edit distances (Damerau, 1964) and normalized by sequence length to make results comparable across sequences of different length. The resulting distance measure, DL_n, see Eq. 2, ranged from 0 (i.e., identical) to 1 (i.e., maximally different). As a baseline, mean edit distance was calculated for $n!$ samples generated without replacement for observed sequences of length n and averaged over all sequences. For each parameter combination, model-generated and observed sequences were compared for $n = 100$ iterations, considering the median edit distance over all iterations.

$$DL_n = \frac{\text{edit distance}}{\text{maximum edit distance}} \tag{2}$$

Using a modeling approach, we investigated planning depth and dimensionality across three table setting datasets (Sec. Data). We estimated k and c by finding the best-fitting model over all unique sequences of action orderings. Values for (strong) k were tested in a range between 0.1 and 0.8 (including ending values), with medium-strength k defined as

$k + 0.1$ and steps of 0.1. Parameter c was tested in the range between 1.1 and 1.9 (including ending values), with steps of 0.1.

Data

TUM Kitchen The TUM Kitchen Data Set (Tenorth, Bandouch, & Beetz, 2009) contains data from four subjects setting a table in different ways, each time using the same items in the same environment. Each trial began with the subject facing the kitchen (standing between location A and B, see Fig. 2) and ended with all required items being on the table (at location C or D). The necessary items for table setting were stored in location A (tray, napkin), in the drawer between A and B (silverware), and B (plate, cup). The x axis represented the traversable space between table and storage locations (cupboards, drawers) as well as kitchen appliances (stove, fridge), while the y axis represented the axis of movement along storage locations and kitchen appliances (fridge, cupboard, stove, etc., see Fig. 2). Of the 20 video episodes, video 18 consists only in repetitive movement and had to be excluded from our analysis.

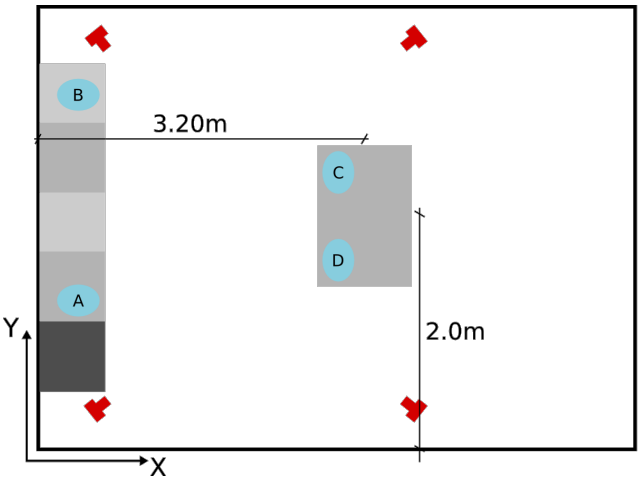


Figure 2: Layout of the TUM kitchen (Tenorth et al., 2009)

EPIC-KITCHENS EPIC-KITCHENS (Damen et al., 2018) is a large-scale first-person vision data set collected by 32 participants in their native kitchens. Since each participant recorded their activities in their home kitchen, spatial environments and items vary between participants.

The participants recorded all their daily kitchen activities with a head-mounted GoPro (video and sound) for three consecutive days. Each recording starts with the participant entering the kitchen and stops before leaving the kitchen. The participants were asked to be in the kitchen alone, so that the videos capture only one-person activities. Each participant recorded several episodes.

The episodes contain a multitude of kitchen activities, such as cooking, stowing away groceries, and table setting. For the purpose of this analysis, we only used episodes with table setting actions, which reduced the sample size to 16 videos.[1]

Since the table setting actions are interleaved with cooking actions, specific items can fulfill different functions, such as a plate being used as container for a meal or as an empty (eating) plate. To account for such differences, items are not categorized according to item type but function (e.g., a plate not serving as the eating plate is not considered to have strong relational dependencies as defined in factor k).

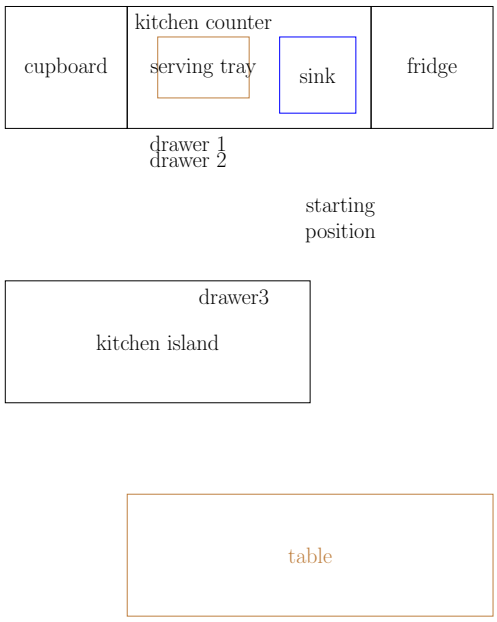


Figure 3: Layout of the Virtual Reality kitchen

Virtual Reality Dataset The data contains table setting sequences in a VR environment from a single participant.The virtual kitchen consisted of three separate regions (fridge, tray area, island area; Fig. 3), each of which had to be visited at least once. The fridge contained a number of dairy products and orange juice, drawer 1 silverware, drawer 2 mugs and glasses, drawer 3 bowls, and the cupboard a number of food packages, such as cereal. The participant moved through the virtual environment by moving through a corresponding but open physical space, experiencing the virtual environment through a HTC Vive head-mounted display. Movement was tracked via the head-mounted display while interaction with the environment was realized through two HTC Vive controllers (one in each hand).

The participant was asked to set the table for one person having breakfast. The minimum set of items (cereal bowl, spoon, cereal, milk, glass, juice) could be expanded by the participant if desired. The task was to first assemble all necessary items on the tray and then to carry the items to the table. The participant was familiar with the kitchen and knew

[1] P01_01, P01_03, P01_05, P01_09, P10_01, P12_01, P12_06, P21_01, P21_03, P21_04, P22_12, P22_16, P24_02, P24_04, P24_05, P26_11.

the location of all required items well. Data from 39 trials was collected. For action orderings we considered the order in which items were grasped and put on the tray.

Model Comparisons

Planning Depth

We ran model simulations for one and two steps of planning ahead (Fig. 4). The one-step model works as described above. The two-step model works as follows: after choosing a first item, a second item is already chosen while picking up the first item, based on the same weighted cost calculation as before. The second item is then picked up next regardless of whether it is the lowest-cost item for the next starting point, repeating this process until task completion. Because both models have the same number of parameters, functional form, and draw on identical sample sizes, a goodness of fit measure is equivalent to more complex measures of generalizability (Pitt & Myung, 2002). Accordingly, we considered and report goodness of fit measures for comparing the models.

Both models consider a 3D environment for distance calculation (see Eq. 1). The best fit for the one-step model is achieved for parameters strong $k = 0.6$, medium $k = 0.7$, and $c = 1.9$, which yield an average edit distance of 0.411 (median: 0.4). The best fit for the two-step model is achieved for parameters strong $k = 0.5$, medium $k = 0.6$, and $c = 1.2$, which yield an average edit distance of 0.415 (median: 0.4). Both results are lower than the baseline of 0.603 (see Simulations).

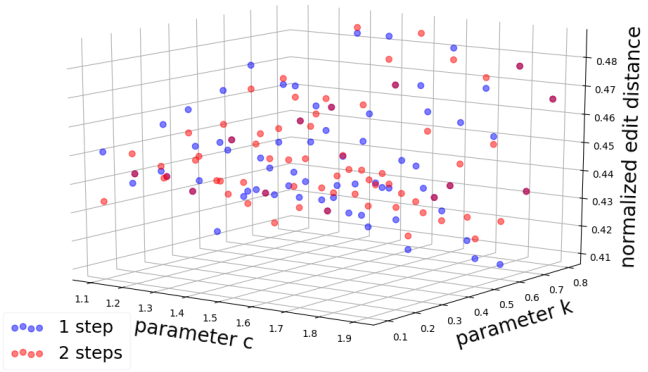


Figure 4: Model fit based on planning depth (k = strong k)

Although the models differ in the action orders they generate, they seem to perform similarly well in accounting for human behavior (Fig. 4). To further investigate, we computed the average edit distances across all possible parameter value combinations. Again, models performed very similar (1 step: 0.446, 2 steps: 0.446, median for both: 0.4) and comparing their prediction accuracy using the Wilcoxon signed rank test shows no significant difference ($W = 831.500$, $p = 0.686$).

As cognitive offloading tends to be used particularly often in spatial tasks (Wilson, 2002), we argue that the results indicate that people plan only one step ahead, as the process of remembering the second item to be picked up can be considered cognitively effortful and adding a second step does not achieve a better fit between predicted and observed behavior.

Dimensionality

To assess dimensionality, we compared seven models that assumed spatial representations along the x, y, z, xy, xz, yz, xyz axes, respectively, all of which assumed one-step planning. For the same reasons as with the depth model comparison, we again used goodness of fit as comparison measure.

Prediction accuracies for the first simulation show a highly significant difference ($\chi^2(6) = 507.748$, $p < 0.001$), which lends support to the idea that dimensionality has a strong influence on action organization in everyday activities. Since previous research shows a preference for 2D spatial representation and better navigation performance in 2D environments, we assume that calculating distances in 2D instead of 3D might reduce the necessary cognitive effort.

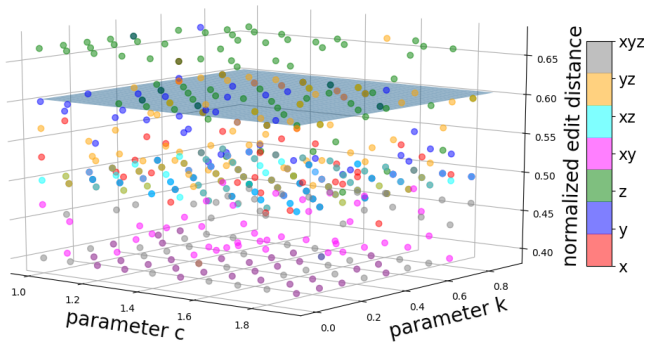


Figure 5: Model fit based on dimensionality (k = strong k)

The distribution shows that the average edit distance between model-generated and observed sequences is lowest when considering xy or xyz for dimensionality (Fig. 5, baseline shown as plane), with xy achieving slightly better results (0.438 vs. 0.444, averaged over all possible parameter combinations; median for both 0.4). In a pairwise comparison of model simulations based on xy and xyz spatial representations using the Wilcoxon signed rank test, the model results considering a horizontal versus a horizontal and vertical spatial representation also differ significantly ($W = 1561.000$, $p = 0.05$), indicating that people seem to ignore the vertical dimension.

As the importance of single (1D) axes might be dependent on how much they can influence the calculation of physical distance, i.e., the actual possible movement span, we compared the span for each axis (x, y, z). y has the highest average span: 3.17 vs. 1.89 and 1.833 for x and z, respectively. The average edit distance and the average volume of all task environments show a strong negative correlation (ρ = -0.708, $p < 0.001$), i.e., with decreasing volume/span of the task environment, the prediction error increases (Fig. 6).

In order to account for the possibility that people assign different importance to the individual spatial axes dependent on their span width, we ran a second simulation of the model

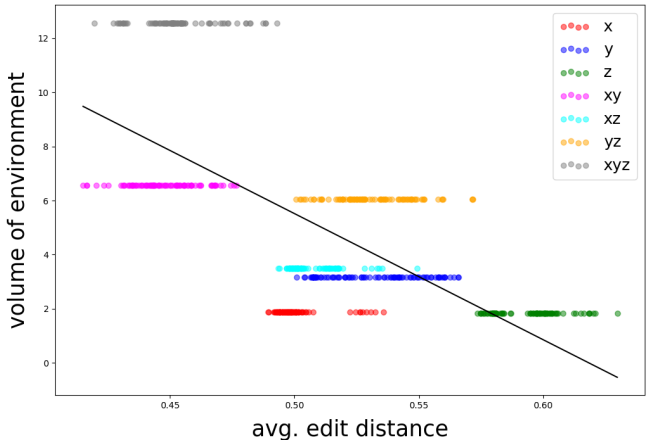


Figure 6: Correlation between average edit distance and volume/span of task environments for each spatial representation

that incorporated a weight criterion for each axis. We calculated a weighted Euclidean distance between item locations by multiplying the partial difference for each axis as shown in Eq. 3, where the axis weight w_n was defined as in Eq. 4. Assuming an environment with axes spans $x = 3$, $y = 2$ and $z = 1$, this results in $w_x = 0.5$, $w_y = \frac{1}{3}$, and $w_z = \frac{1}{6}$.

$$d(p,q) = \sqrt{(p_x - q_x)^2 \cdot w_x + (p_y - q_y)^2 \cdot w_y + (p_z - q_z)^2 \cdot w_z} \tag{3}$$

$$w_x = \frac{span_x}{span_x + span_y + span_z} \tag{4}$$

In the new model, z still shows the highest error rate in prediction (0.60 average edit distance, thus similar to the baseline), whereas xyz achieves a slightly better fit, but still has a higher average edit distance than xy (xyz: 0.510, xy: 0.514; median: 0.51 for both). Comparing two- and three-dimensional representation using the Wilcoxon signed rank test indicates a significant difference ($W = 1348.000$, $p = 0.005$), i.e., xy achieves the best fit in both model variations.

Conclusion and Future Work

Our results lend to support to our initial argument of the merits of investigating cognitive abilities by analyzing everyday activities. Our analyses of table setting provided two main findings: First, people behave consistently with a model that plans one step ahead and, second, a representation of two-dimensional horizontal space seems to be preferred over a three-dimensional representation including the vertical.

Both findings indicate that the cognitive costs of alternative planning strategies and representation structures outweigh their potential benefits. These findings are consistent with previous research showing human navigation performance to be better in 2D environments (Zwergal et al., 2016), differences in the accuracy of distance encoding in horizontal vs. vertical space (Hinterecker et al., 2018), and the theories of external scaffolding and cognitive offloading, i.e., humans using properties of the environment to their advantage.

We expect our proposed planning model not to be specific to the task of table setting, but to be generalizable to other everyday activities as well. Aspects to consider in future models are possible interdependency effects between planning depth and dimensionality, as well as cognitive effort. While we consider cognitive effort in the scope of relational dependencies and dimensionality, further research is needed on how cognitive effort impacts everyday activities.

As the model is not able to provide reliable predictions for sequences with low variance in the considered constraints (e.g., similar distances, no relational dependencies between items or containment), other potentially influential factors need to be investigated further and addressed in future versions.

Acknowledgments

The research reported in this paper has been supported by the German Research Foundation DFG, as part of Collaborative Research Center (Sonderforschungsbereich) 1320 “EASE - Everyday Activity Science and Engineering”, University of Bremen (http://www.ease-crc.org/). The research was conducted in subproject P03 “Spatial Reasoning in Everyday Activity”.

References

Botvinick, M., & Plaut, D. C. (2004). Doing without schema hierarchies. *Psychological Review*, *111*(2), 395–429. doi: 10.1037/0033-295X.111.2.395

Clark, A. (1996). *Being there. Putting brain, body, and world together again*. MIT Press.

Cooper, R. P., & Shallice, T. (2006). Hierarchical schemas and goals in the control of sequential behavior. *Psychol Rev*, *113*(4), 887–916. doi: 10.1037/0033-295X.113.4.887

Damen, D., Doughty, H., Farinella, G. M., Fidler, S., Furnari, A., Kazakos, E., ... Perrett, T. (2018). Scaling Egocentric Vision: The EPIC-KITCHENS Dataset. In *The European Conference on Computer Vision (ECCV)* (pp. 720–736).

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171–176. doi: 10.1145/363958.363994

Hinterecker, T., Pretto, P., de Winkel, K. N., Karnath, H.-O., Bülthoff, H. H., & Meilinger, T. (2018). Body-relative horizontal–vertical anisotropy in human representations of traveled distances. *Experimental Brain Research*, *236*(10), 2811–2827. doi: 10.1007/s00221-018-5337-9

Hull, C. L. (1943). *Principles of behavior: An introduction to behavior theory*. Oxford, England: Appleton-Century.

Icard, T. F. (2018). Bayes, bounds, and rational analysis. *Philosophy of Science*, *85*(1), 79–101.

Jeffery, K. J., Jovalekic, A., Verriotis, M., & Hayman, R. (2013). Navigating in a three-dimensional

world. *Behavioral and Brain Sciences*, *36*(5), 523–543. (Publisher: Cambridge University Press) doi: 10.1017/S0140525X12002476

Jones, M., & Love, B. C. (2011). Bayesian fundamentalism or enlightenment? On the explanatory status and theoretical contributions of bayesian models of cognition. *Behavioral and Brain Sciences*, *34*(4), 169–188.

Kirsh, D. (1995). The intelligent use of space. *Artificial Intelligence*, *73*(1), 31–68.

Kool, W., McGuire, J. T., Rosen, Z. B., & Botvinick, M. M. (2010). Decision making and the avoidance of cognitive demand. *Journal of Experimental Psychology*, *139*(4), 665–682.

Meder, B., Nelson, J. D., Jones, M., & Ruggeri, A. (2019). Stepwise versus globally optimal search in children and adults. *Cognition*, *191*. doi: 10.1016/j.cognition.2019.05.002

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In *Visual Information Processing: Proceedings of the Eighth Annual Carnegie Symposium on Cognition* (pp. 283–308). Academic Press.

Pitt, M. A., & Myung, I. J. (2002). When a good fit can be bad. *Trends in Cognitive Sciences*, *6*(10), 421–425. doi: 10.1016/S1364-6613(02)01964-2

Schurz, G., & Thorn, P. D. (2016). The revenge of ecological rationality: Strategy-selection by meta-induction within changing environments. *Minds and Machines*, *26*(1), 31–59.

Simon, H. A. (1955). A behavioral model of rational choice. *The Quarterly Journal of Economics*, *69*(1), 99–118.

Tenorth, M., Bandouch, J., & Beetz, M. (2009). The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition. In *IEEE International Workshop in conjunction with ICCV2009.*

van de Ven, J., Fukuda, M., Schultheis, H., Freksa, C., & Barkowsky, T. (2018). Analyzing strong spatial cognition: A modeling approach. In S. Creem-Regehr, J. Schöning, & A. Klippel (Eds.), *Spatial Cognition XI* (pp. 197–208). Springer International Publishing.

Wenzl, P., & Schultheis, H. (2020). Optimality and space in weakly constrained everyday activities. In *Proceedings of the 42nd Annual Meeting of the Cognitive Science Society.* (accepted)

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, *9*(4), 625–636.

Zwergal, A., Schöberl, F., Xiong, G., Pradhan, C., Covic, A., Werner, P., ... Brandt, T. (2016). Anisotropy of Human Horizontal and Vertical Navigation in Real Space: Behavioral and PET Correlates. *Cerebral Cortex*, *26*(11), 4392–4404. doi: 10.1093/cercor/bhv213

An Expanded Set of Declarative Memory Functionalities in PyACTUp, a Python Implementation of ACT-UP's Accountable Modeling

Yuxue C. Yang (chery@uw.edu)
Department of Psychology, University of Washington
Campus Box 3515525, Seattle, WA 98195 USA

Don Morrison (dfm2@cmu.edu)
Department of Psychology, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213

Andrea Stocco (stocco@uw.edu)
Department of Psychology, University of Washington
Campus Box 3515525, Seattle, WA 98195 USA

Mark Orr (mo6xj@virginia.edu)
Biocomplexity Institute, University of Virginia,
P.O. Box 400298, Charlottesville, VA 22904 US

Christian Lebiere (cl@cmu.edu)
Department of Psychology, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213

Keywords: Cognitive Architecture; Accountable Modeling; ACT-R; ACT-UP; Python; Agile Development

Introduction

ACT-R (Anderson, 2007) is the most influential cognitive architecture in psychology and neuroscience (Kotseruba and Tsotsos, 2018). To enforce its commitment on *end-to-end* modeling (that is, the constraint that the modeler should consider *all aspects* of a task), ACT-R is purposely designed to paradigmatically discourage the developer from taking shortcuts, which is often frustrating.

In 2010, Reitter and Lebiere proposed the alternative paradigm of *accountable modeling*. Instead of discouraging developers from taking shortcuts, it encourages them to explicitly specify which components of an architecture they plan to use and how they account for aspects that are not modeled using the architecture (e.g., through parameter estimation). As a proof of concept, they designed ACT-UP, a modular Lisp implementation of ACT-R's declarative memory system that can be used independently of the other components. Despite its successes, ACT-UP was limited by (a) being written in Lisp, (b) covering only a limited set of functions; and (c) inheriting ACT-R's traditional assumption that memories are equal in terms of importance.

Here, we present PyACTUp, a Python implementation of ACT-UP that takes advantage of Python's modern language design, and extensive graphic and scientific libraries, and further extends the set of declarative memory functionalities, including spreading activation and emotional components. Both of these core functionalities have received widespread attention in contemporary models, and are now implemented without any reference to ACT-R's other first-level structures (buffers or productions). Github: https://cher_yang@bitbucket.org/cher_yang/pyactup2.git

General PyACTUp Architecture

PyACTUp closely follows Reitter & Lebiere's (2010) original implementation of ACT-UP, doing away with buffers and procedural knowledge and simplifying the process of modeling memory encoding and retrieval, so that modelers are able to focus on essential cognitive phenomena instead of being stopped by programming difficulties.

The most fundamental functions of PyACTUp are learning and retrieving memories. As pieces of information are *learned*, they are stored in declarative memory as *chunks*. The memory contains one or more *slot-value* pairs, and can be *retrieved* by specifying all or a subset of identifying *cues,* also in the form of slot-value pairs. A new declarative memory is created as an instance of the `Memory` class object, i.e. `dm = Memory()`. Chunks are represented using Python's built-in dictionary type, and are learned and retrieved using Python's keyword-argument syntax to manage slot-value pairs. For example, the following line:

```
dm.learn(guy="Ringo",role="Drummer")
```

creates a chunk for Ringo Starr of The Beatles. To retrieve him, one can type

```
dm.retrieve(role="Drummer")
```

As in ACT-R, chunks in a `Memory` structure are retrieved based on their *activation*, a scalar meta-quantity that reflects frequency (the retrieval probability odds) and the recency (decays over time). Depending on the number of memories available in a `Memory` structure, it is possible to enable *blending* and retrieve *blended* memories instead of a specific chunk.

Implementation of Spreading Activation

In ACT-R, a chunk's activation is made of a *base-level* (the part that decays over time) and a *spreading* component, which increases a chunk's activation in proportion to its association to other chunks. Because spreading activation originates in ACT-R buffers, ACT-UP and its original Python counterpart originally excluded it from their available functionalities. However, since spreading activation is important to capture phenomena like the fan effect (Anderson, 1974) and working memory (Daily et al 2001), it was introduced as new functionality in this version of PyACTUp. To avoid the use of buffers, activation is spread through a specific function and the use, again of the

keyword-argument syntax, with an argument that defines the source. For example,

```
dm.spread(role="drummer")
```

spreads activation from "Drummer" to other chunks in `dm`. Following the ACT-R standard algorithm, the associative strength between two chunks *i* and *j* is calculated, by default, as inversely proportional to the logarithm of the fan of *i*, i.e. the number of times *i* appears in other chunks other than *j*. Modelers, however, are given flexibility to define their own associative strength functions. The value returned by these functions is then multiplied by the weight.

Differential Importance of Memories

ACT-R and ACT-UP share the assumption that all memories are equal in terms of importance, and therefore they decay at the same rate. This assumption was recognized as a limitation by Anderson himself (2007, chapter 3). Researchers modeling effects of emotion, for example, have repeatedly suggested that affect and emotion alter how memorable certain events are by providing an activation bias or boost (Juvina, Larue, & Hough, 2018, Fum & Stocco, 2004; Cochran et al., 2006).

Without taking a stance in the debate, we decided to provide a way for developers to add their own activation bias terms through an additional importance term, which is linearly additive to base-level and spreading activation. Here, we implement the highest level of conceptual idea of importance. The degree of importance is attached to the nature of memory and to some extent decides the needs of retrieval subsequently. The importance is set directly through learning. For example,

```
dm.learn(guy="Disco",role="Manager",
importance=5)
```

To observe the effects of importance on memory retrieval, we created a model to simulate the retrieval process of memory with various degrees of importance. Highly important memory is set to values from 0 to 3 while normal memory is set to a randomly distributed value from 0 to 2.

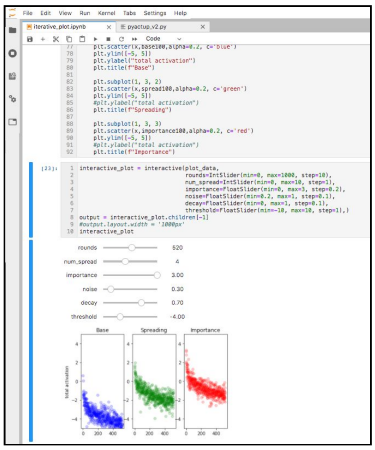


Fig 1. *Interactive Model shows the change of base-level activation over time (blue); base-level + spreading (green); and base-level + spreading + importance (red).*

Interaction Example

Fig 1 provides an example of the code in Jupyter Notebook window. This shows how PyACTUp can be used to quickly run an interactive model in Python. By adjusting parameters, the model shows dynamic memory encoding and retrieving outputs.

Summary

The current version of PyACTUp inherits the simplification from the last version and extends its functionality to include the spreading activation and importance term. We use the new PyACTUp to model spreading activation effects and the retrieval discrepancy between important memory and trivial ones. Admittedly, PyACTUp is only a subset of the ACT-R architecture, so it does have some limitations in modeling human cognition. How to keep PyACTUp simple, flexible and applicable is a trade-off problem for the framework designer. Further researchers could explore other useful functions and expand the current PyACTUp to a more comprehensive, accountable and modular implementation of the ACT-R cognitive modeling architecture.

References

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology, 6*(4), 451–474

Anderson, J. R. (2007). How can the human mind occur in the physical universe? *Oxford University Press*.

Chown, E., Cochran, R. E, & Lee, F. J. (2006). Modeling emotion: Arousal's impact on memory. *Proceedings of the Annual Meeting of the Cognitive Science Society, 28.*

Daily, L. Z., Lovett, M. C., & Reder, L. M. (2001). Modeling individual differences in working memory performance: A source activation account. *Cognitive Science, 25*(3), 315-353.

Fum, D., & Stocco, A. (2004). Memory, emotion, and rationality: An ACT-R interpretation for gambling task results. In C. D. Schunn, M. C. Lovett, C. Lebiere & P. Munro (Eds.) *Proceedings of the 6th International Conference on Cognitive Modeling.* Mahwah, New Jersey: Lawrence Erlbaum Associates, pp. 106–111.

Juvina, I., Larue, O., & Hough, A. (2018). Modeling valuation and core affect in a cognitive architecture: The impact of valence and arousal on memory and decision-making. *Cognitive Systems Research, 48*, 4-24.

Kotseruba, I., & Tsotsos, J. K. (2018). 40 years of cognitive architectures: Core cognitive abilities and practical applications. *Artificial Intelligence Review*, 1-78.

Reitter, D., & Lebiere, C. (2010). Accountable modeling in ACT-UP, a scalable, rapid-prototyping ACT-R implementation. In *Proceedings of the 10th International Conference on Cognitive Modeling, ICCM 2010* (pp. 199-204)