Proceedings of ICCM 2018

16th International Conference on Cognitive Modeling[1]

July 21-24, 2018

University of Wisconsin, Madison, WI.

Edited by

Ion Juvina, Joseph Houpt, and Christopher Myers

[1] Collocated with The 51st Annual Meeting of the Society for Mathematical Psychology at the University of Wisconsin in Madison, WI.

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 modeling 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 2018, ICCM was jointly held with MathPsych – the annual meeting of the Society for Mathematical Psychology at the University of Wisconsin, in Madison, WI, USA, on July 21st-24th.

Acknowledgments

We would like to acknowledge the Society for Mathematical Psychology (SMP), the Department of Psychology at the University of Wisconsin, Madison, and Springer, whose combined support kept the conference fees low and allowed us to fund a number of student awards. We also would like to acknowledge the people who brought the MathPsych and ICCM conferences together for the first time (Andrew Heathcote, Amy Criss, Frank Ritter, and David Reitter), the hard work of the MathPsych local organizer (Joe Austerweil), the officers of the SMP (Brent Miller, Leslie Blaha, Jennifer Trueblood, Scott Brown, and Pernille Hemmer), and University of Wisconsin conference services for their logistical support. EasyChair was used to manage submissions and reviews and generate this volume.

Papers in this volume may be cited as:

LastNameAuthorA, FirstInitialA., LastNameAuthorB, FirstInitialB., & LastNameAuthorC, FirstInitialC. (2018). This is the title of the paper. In I. Juvina, J. Houpt, & C. Myers (Eds.), *Proceedings of the 16th International Conference on Cognitive Modeling* (pp. 6-12). Madison, WI: University of Wisconsin.

ISBN-13: 978-0-9985082-2-1

Conference Committees

General and Program Chairs

Ion Juvina, Wright State University

Joseph Houpt, Wright State University

Christopher Myers, The United States Air Force Research Laboratory

Program Committee

Erik	Altmann	Michigan State University
Adrian	Banks	University of Surrey
Thomas	Barkowsky	University of Bremen
Leslie	Blaha	Pacific Northwest National Laboratory
Jelmer	Borst	University of Groningen
Mike	Byrne	Rice University
Richard	Carlson	Penn State Psychology
Christopher	Dancy	Bucknell University
Justin	Estepp	Air Force Research Laboratory
Francesco	Gagliardi	Italian Association for Cognitive Sciences
Moojan	Ghafurian	University of Waterloo
Kevin	Gluck	Air Force Research Laboratory
Fernand	Gobet	University of Liverpool
Glenn	Gunzelmann	Air Force Research Laboratory
Bill	Kennedy	George Mason University
David	Kieras	University of Michigan
Johan	Kwisthout	Radboud University
Ralf	Mayrhofer	University of Göttingen
Krishna Prasad	Miyapuram	Indian Institute of Technology Gandhinagar
Junya	Morita	Shizuoka University
Shane	Mueller	Michigan Technological University
David	Noelle	University of California, Merced
Enkhbold	Nyamsuren	The Open University
David	Peebles	University of Huddersfield
Nele	Russwinkel	Technische Universitat Berlin
Dario	Salvucci	Drexel University
Ute	Schmid	University of Bamberg
Michael	Schoelles	Rensselaer Polytechnic Institute
Jennifer	Spenader	University of Groningen
Robert	St. Amant	North Carolina State University
Christopher	Stevens	Air Force Research Laboratory
Terrence	Stewart	University of Waterloo
Andrea	Stocco	University of Washington
Ron	Sun	Rensselaer Polytechnic Institute
Greg	Trafton	Navy Research Laboratory
Marieke	van Vugt	University of Groningen
Sharon	Wood	University of Sussex
Iraide	Zipitria	The University of the Basque Country

Table of Contents

An extensible framework for mechanistic processing models: From representational linguistic theories to quantitative model comparison

Adrian Brasoveanu (abrsvn@gmail.com)
Department of Linguistics, 1156 High Street
Santa Cruz, CA 95065, USA

Jakub Dotlačil (j.dotlacil@gmail.com)
ILLC, Science Park 107
1098 XG Amsterdam, The Netherlands

Abstract

We introduce a Python3 reimplementation of ACT-R (Anderson and Lebiere 1998, Anderson 2007) in which we build an end-to-end simulation of syntactic parsing in a typical self-paced reading experiment. The model uses an eager left-corner parsing strategy implemented as a skill in procedural memory (following Lewis and Vasishth 2005), makes use of independently motivated components of the ACT-R framework (procedural memory, content-addressable declarative memory – cf. Wagers et al. 2009), and explicitly models the motor and visual processes involved in self-paced reading. The ACT-R model can be embedded in a Bayesian statistical model to estimate its subsymbolic parameters and perform model comparison.

Keywords: ACT-R, Bayesian models, incremental processing, syntax, semantics, self-paced reading

Introduction: framework & case study

The overarching goal of the research we report on here is to build an extensible framework in which formally and computationally explicit processing models for natural language syntax and semantics can be formulated. Specifically, we want to build cognitively realistic models for incremental parsing of discourse representations structures (DRSs, Kamp 1981; Kamp and Reyle 1993) or similar representations. In the models we built so far, the semantic and syntactic representations are created mostly in parallel, so we will be able to focus only on modeling *syntactic* representations in this paper.

This extensible framework enables us to formulate *mechanistic* models of natural language processing, which is the preferred level of explanation in cognitive science (see Lewandowsky and Farrell 2010 among many others). When building the framework, our strategy was to use an independently motivated, general cognitive architecture, and to embed processing models for natural language in this architecture. Since parsing is easy to embed in hybrid (symbolic and subsymbolic) cognitive architectures, we chose to focus on them. Two hybrid architectures are in common use in psycholinguistics, namely Soar (Hale 2014; Young and Lewis 1999) and ACT-R (Dillon et al. 2013; Engelmann et al. 2013; Kush 2013; Lewis and Vasishth 2005; Nicenboim and Vasishth 2018; Rij 2012; Taatgen and Anderson 2002; Vasishth et al. 2008). ACT-R is the more popular architecture, so it was a natural choice for our framework.

Currently, psycholinguistic ACT-R models are mainly used to model recall of syntactic structures (Dillon et al. 2013; Engelmann et al. 2013; Lewis and Vasishth 2005; Nicenboim and Vasishth 2018; Vasishth et al. 2008). This focus on recall-related modeling does not take advantage of the generality of ACT-R as a cognitive architecture and its "no magic" policy. If we want to make explicit all the various parsing components and actions involved in processing models for specific natural language phenomena, we have to rely on the full implementation of ACT-R in LISP, which is not a very popular programming choice now. Furthermore, since LISP is a relatively isolated programming language, it does not have a thriving ecosystem of statistical estimation / machine learning libraries that could be leveraged in processing models. Being a cognitive architecture, ACT-R comes with many parameters, but because of the relative paucity of the LISP library ecosystem, these parameters are set to

their default values or manually changed even when LISP ACT-R is used. Manually changing parameters makes modeling hard to replicate and systematic quantitative model comparison hard to perform.

In this paper, we introduce a new Python3 implementation of ACT-R (**pyactr**, Brasoveanu and Dotlačil 2018, in prep.), which makes two main contributions. On one hand, it is easy to combine ACT-R modeling and Bayesian estimation methods: ACT-R models are embedded in Bayesian models, which makes it possible for us to systematically explore parameter values and quantify our uncertainty about them, perform quantitative model comparison, and replicate / build on previous modeling work. On the other hand, the ACT-R component comes with a working, extensible parsing framework for syntax and semantics, which includes visual and motor interfaces and a variety of models for both syntactic and semantic phenomena. The framework has a modular structure: alternative models for peripherals (visual, motor) and other components can in principle be swapped in, and the resulting models can be systematically compared.

We showcase the framework by modeling Experiment 1 in Grodner and Gibson (2005) (also used in Lewis and Vasishth 2005). This is a self-paced reading experiment (non-cumulative moving-window; Just et al. 1982). Participants read word-by-word sentences in which the subject noun phrase (NP) is modified by a subject or object extracted relative clause (RC). A subject-gap example is provided in (1), and an object-gap in (2).

(1) The reporter who GAP sent the photographer to the editor hoped for a story.

(2) The reporter who the photographer sent GAP to the editor hoped for a story.

There are 9 regions of interest (ROIs) that we will model, underlined in the examples above. These are word 2 (the matrix noun in subject position) through word 10 (the matrix verb). An ACT-R model for these 2 sentences is demo-ed in Figure 1. The red circle is the visual focus; the models goes through a series of cognitive (parsing) steps and decides to press the space bar to reveal the next word at certain times during this process. The temporal trace incrementally produced by the model (with all interacting modules, buffers, parsing actions etc.) is visible in the background.

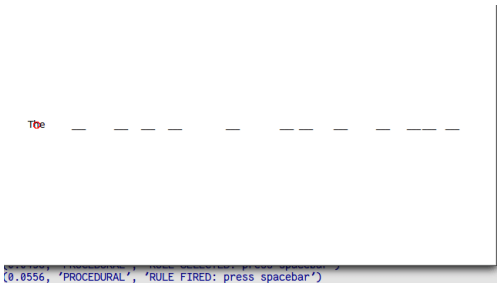


Figure 1: An ACT-R model for self-paced reading (open the paper with Adobe Reader to see movie)

The remainder of this paper is dedicated to unpacking this ACT-R model (**ACT-R and eager left-corner parsing**) and quantitatively comparing three variations on it that differ in several theoretically-relevant ways (**Modeling results**). We conclude with a summary and future research directions.

ACT-R and eager left-corner parsing

We unpack the ACT-R model of self-paced reading demo-ed in Figure 1 at three different levels of detail. We start with a broad overview of the ACT-R architectural components we need. We then outline how various parts of an eager left-corner parser are distributed over the ACT-R components. Finally, we discuss how the model functions on a per-word basis; that is, we outline the cognitive steps the model goes through starting immediately after a word is revealed on the virtual screen and ending with the point at which the model decides to press the space bar to reveal the next word.

There are two types of memory in ACT-R: (i) declarative memory (roughly, 'knowing that') – knowledge of facts, which are represented as chunks (attribute-value matrices), e.g., the lexical chunk for the word *car* in (3); and (ii) procedural memory (roughly, 'knowing how') – the set of productions that fire in series to generate cognitive behavior / processes. These productions have the form of rewrite rules in formal grammars (e.g., context free / phrase structure grammars), but in ACT-

R, they are conditionalized cognitive actions: the ACT-R mind fires a production, i.e., takes the action encoded in it, if the current cognitive state satisfies the preconditions of that production.

(3)

ISA:	word
FORM:	car
MEANING:	⟦car⟧
CATEGORY:	noun
NUMBER:	sg

(4)

Goal>	TASK:	reading	
	FORM:	car	

⇒

Goal>	TASK:	retrieving category	
Retrieval>	ISA:	word	
	FORM:	car	

An example production is provided in (4):

- if the current cognitive state is such that the goal buffer (which drives cognitive processes in ACT-R) encodes a TASK of 'reading' the FORM 'car',
- then (⇒) we take a cognitive action that takes us to a new cognitive state,
- where the TASK is to retrieve the syntactic category of that form, and in which we simultaneously place a request in the Retrieval buffer to search declarative memory for a chunk of type 'word' that has the FORM 'car'.

Implicit in this example production is that an ACT-R mind is composed of modules, which include declarative and procedural memory, but also visual and motor modules etc. Modules are not directly accessible: they can only be accessed through associated buffers (e.g., the Retrieval buffer is associated with declarative memory). Buffers serve a dual purpose: individually, they provide the input/output interface to specific modules; as a whole, however, buffers represent the current cognitive state of the mind. Crucially, productions fire based on the current cognitive state, i.e., conditioned on the contents of various buffers. The ACT-R architecture constrains cognitive behavior in various ways, two of which are that buffers can hold only one chunk, and only one production can fire at any given time.

Let us now move on to how we can implement an eager left-corner parser in ACT-R (building on Lewis and Vasishth 2005; Resnik 1992; see also Hale 2014 for an introduction). We distribute parser components over ACT-R modules and buffers as follows. Lexical knowledge is encoded in declarative memory, knowledge of grammar and parsing actions are encoded in procedural memory, expectations about upcoming syntactic categories (which guide parsing) are encoded in the goal buffer, information about the current partially-built syntactic parse is encoded in the imaginal buffer (a secondary goal-like buffer), visual information from the environment is transferred via the visual buffer, and, finally, key press commands are issued via the manual buffer. The visual module we implement is EMMA (Salvucci 2001), and the motor module is EPIC (Kieras and Meyer 1996; Meyer and Kieras 1997). Other choices are also possible.

Running this eager left-corner parser on a simple input sentence will shed more light on its inner workings and how they are deployed in real time. Assume we have a simple grammar with three phrase structure rules S → NP VP, NP → Det N, and VP → V. Also, assume that we are reading the sentence *A boy sleeps* in a self-paced reading task.

We start with a screen in which all words are covered with dashes: `- --- ------`. Our goal stack (stored in the goal buffer) consists of just S: our goal is to parse a sentence. After the first space-bar press, the first word is revealed: `A --- ------`, its visual form is transferred via the visual buffer, and its syntactic category Det (determiner) is retrieved from declarative memory. At that point, we take a series of cognitive steps – that is, we fire a series of productions – that take us to a new state. The goal stack in this new state is N NP S: we now have two subgoals of finding an N (noun) and an NP (noun phrase) on the way to finding an S. Also, we build a partial syntactic structure of the form shown in Figure 2 and store it in the imaginal buffer. We see here the left-corner nature of our parser: we trigger all the syntactic rules that have the determiner *a* or

a node dominating *a* as their left branch.

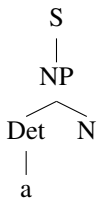


Figure 2: Partial tree after reading the determiner *a*

After another space-bar press, the noun is revealed (`- boy ------`), its form is transferred via the visual buffer and its syntactic category N is retrieved from declarative memory. At this point, we trigger a series of productions that discharge all the N, NP and S goals (this reflects the eager nature of the parser) and replaces them with the single goal of finding a VP (verb phrase). At the same time, a richer partial tree, shown in Figure 3, is stored in the imaginal buffer.[1]

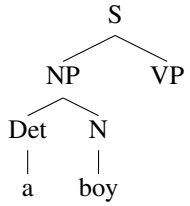


Figure 3: Partial tree after reading the noun *boy*

Finally, the verb is revealed after one more space-bar press: `- --- sleeps`. Its visual form is transferred via the visual buffer and its syntactic category V is retrieved from declarative memory. At that point, the VP goal is satisfied, resulting in an empty goal stack $\varnothing$, and the final structure in Figure 4 is built and encoded in the imaginal buffer.

We have now examined our model at two levels of detail: the general cognitive architecture (ACT-R) and the way the eager left-corner parser is distributed over this architecture. We zoom in one more time to reach the final level of detail at which we want to examine the parser, and describe the series of cognitive steps it takes beginning immediately after seeing a word and ending with the decision to press the space bar to reveal the next word. This sequence of steps is summarized in the flowchart in Figure 5 below.

[1] Strictly speaking, only parts of the tree in Figure 3 are stored in the imaginal buffer at any given time: in the broader spirit of ACT-R, syntactic chunks encode only one level of embedding in the tree, e.g., [$^{\text{NP}}$Det N] or [$^{\text{S}}$NP VP], but not both.

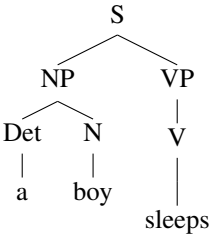


Figure 4: Partial tree after reading the noun *boy*

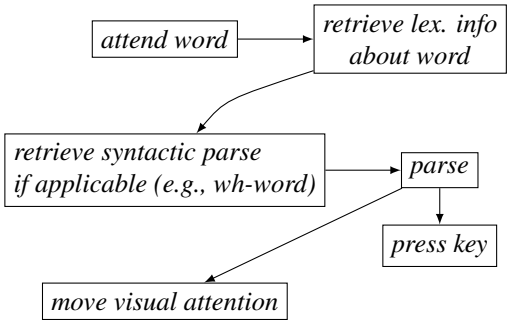


Figure 5: Flowchart of parsing process per word

As this flowchart indicates, we first attend to the visual form of the word, then retrieve lexical information about the word, and also a syntactic structure if applicable (e.g., we retrieve the wh-word at GAP sites). We then proceed with all the parsing actions we can take given our eager left-corner parser. When these parsing actions are complete – *and only then* – we proceed in parallel to moving visual attention and issuing the key-press motor command.

Modeling results

We estimate four different parameters associated with the ACT-R parsing model outlined in the previous section. We could in principle estimate more,

but we confine ourselves to these four parameters for simplicity.

The first one is the angle parameter k that modulates visual encoding; the time of visual encoding T_{enc} is given by the function $K \cdot D \cdot e^{k \cdot d}$, where k is the angle parameter we estimate, d is visual distance, D specifies relevant visual object properties (in our case, word length), and K is set to its default value of 0.01. We estimate this parameter mostly to show that parameters for peripheral modules can be estimated at the same time as the more commonly estimated parameters associated with declarative and procedural memory.

The second one is the time r it takes to fire a production rule (a condition-action pair). This is necessary because our processing models incorporate linguistic theories in a fairly transparent way, which makes it necessary to fire more rules per word / region of interest (ROI) than it would be possible with the 50 ms default. It might be that a judicious use of production compilation will increase rule-firing time closer to its ACT-R default, but this is a topic for future research. Apart from the need to estimate rule-firing time in such 'theoretically-transparent' linguistic applications, the ACT-R + Bayes framework we introduce here enables us to quantify our uncertainty about rule-firing times in *any* ACT-R model; as an anonymous reviewer points out, a good understanding of the uncertainty associated with the r parameter is relevant to ongoing discussions about the need to possibly add noise to it, and will benefit ACT-R models across the board.

The third and fourth parameters are the latency factor F and the latency exponent f that modulate the latency of retrieval from declarative memory. Retrieval latency T is a function of activation A, specifically, $F \cdot e^{-f \cdot A}$.[2] The latency factor F is commonly estimated, but the latency exponent f is usually set to its default value of 1. We estimate both of them here because the latency exponent has proved crucial in estimating latencies in lexical decision tasks like the ones in Murray and Forster (2004) – see Chapter 7 in Brasoveanu and Dotlačil (2018, in prep.) for a detailed discussion. Given that lexical retrieval is a necessary component of any cognitively-realistic parsing model, we estimate both parameters.

The model is fit to data by estimating the posterior distributions of these four free parameters k, r, F and f. Standardly, modelers rely on default values or manually changing the values, but this process is subjective and time consuming (e.g., a grid-based search over only 20 parameter values for just these 4 parameters would require manually evaluating $20^4 = 160000$ combinations). In contrast, **pyactr** enables us to easily interface ACT-R models with standard statistical estimation methods implemented in widely-used Python3 libraries. Specifically, we use ACT-R models as the likelihood component of full Bayesian models (implemented in **pymc3**). We are therefore able to take advantage of much more efficient search methods in multi-dimensional parameter spaces, specifically, Markov Chain Monte Carlo (MCMC) methods, when we fit the ACT-R parameters to experimental data.

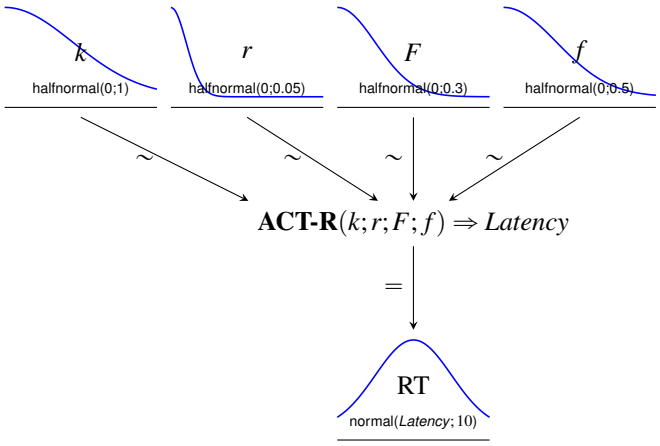


Figure 6: The structure of the Bayesian model

The structure of the Bayesian model is provided in Figure 6: vague / low-information priors for the parameters are listed at the top, the entire ACT-R model provides the likelihood function,

[2] Base activation A is a function of time periods t_k since previous word usages k from 1 to n, where n is determined by the frequency of the word: $A = \log\left(\sum_{k=1}^{n} t_k^{-0.5}\right)$. For reasons of space, we do not discuss spreading activation.

which outputs latencies (times between successive key presses) that can be matched against the reading times (RTs) observed in Grodner and Gibson (2005, Exp. 1). Bayesian methods have many advantages, including the fact that we obtain full posterior distributions for the parameters of interest. We are therefore able to find good parameter values for hybrid (symbolic & subsymbolic) models, and also to quantify our uncertainty about these values. Posterior estimates: k – mean=0.87, sd=0.32, 95% HPD $[0, 1.23]$; r – mean=0.02, sd=0.006, 95% HPD $[0.01, 0.03]$; F – mean=0.01, sd=0.03, 95% HPD $[0, 0.1]$; f – mean=0.23, sd=0.47, 95% HPD $[0, 1.34]$.

The fit of the model is most easily evaluated by examining its posterior predictions for the 9 ROIs, plotted in Figure 7. The diamonds indicate the observed mean RTs for each word, the segments provide the 95% CRIs (credible intervals) for the mean RT predicted by our model, and the dots are the predicted mean RTs. When evaluating the model, recall that the parameters are estimated once for a full run through the experiment – they are not estimated ROI by ROI (falsely assuming independence between ROIs), as it is usually done in the psycholinguistic literature. That is, we model here in one go the full, hybrid (symbolic and subsymbolic) stochastic process of incremental parsing in a self-paced reading task.

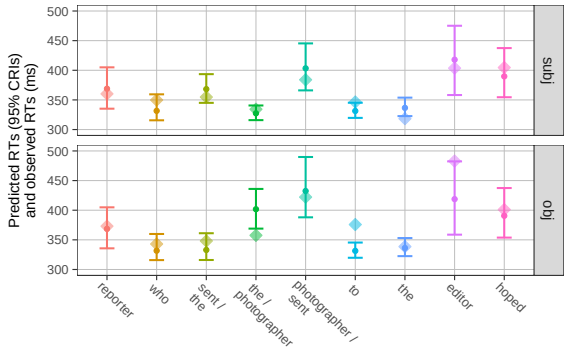


Figure 7: Model 1: postulated subject gaps

Figure 7 shows that wh-gap retrieval is modeled well: this is the 3rd word (*sent*) in the top panel (**subj**-gaps) and the 5th word (also *sent*) in the bottom panel (**obj**-gaps). But the spillover effect on the word after the object gap – the 6th word (*to*) in the bottom panel – is not captured; we return to this.

Finally, we see that the wh-word and the following word (the 2nd and 3rd words in both panels) are modeled well for both subject and object gaps. This is particularly interesting because the model is formulated so that it predictively postulates a subject gap when parsing the wh-word. This postulated gap has to then be reanalyzed for object gaps.

We therefore consider a second model that does not postulate a subject gap as soon as the wh-word is parsed. The posterior predictions of this model, provided in Figure 8, are clearly worse: the 95% CRIs are completely below the observed mean RTs for the wh-word in both conditions, and also for the word immediately following the wh-word in the object-gap condition. This indicates that the model underestimates the parsing work triggered by the wh-word, and it also underestimates the reanalysis work that needs to be done on the word immediately following the wh-word in the object-gap condition.

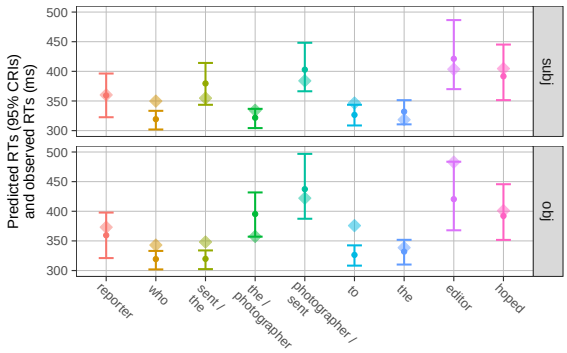


Figure 8: Model 2: no postulated subject gaps

Thus, our first model (postulated subject gaps; Figure 7) is the better one. This model comparison shows that our ACT-R + Bayes framework has empirical bite; not everything goes. Furthermore, the

framework can be used both for formalizing (symbolic) processing hypotheses and for quantitative hypothesis testing.

A final, third model we consider aims to capture the spillover effect on the word following the retrieval of object gaps (the 6th word in the bottom panel of Figure 7 above). In this model, parsing actions proceed in parallel to moving visual attention and issuing key press commands, unlike the flowchart in Figure 5 where we see that parsing has to always precede visual and motor actions. This is sufficient to capture the spillover effect for object gaps, and also increases the precision of our model (smaller CRIs), as shown in Figure 9 below.

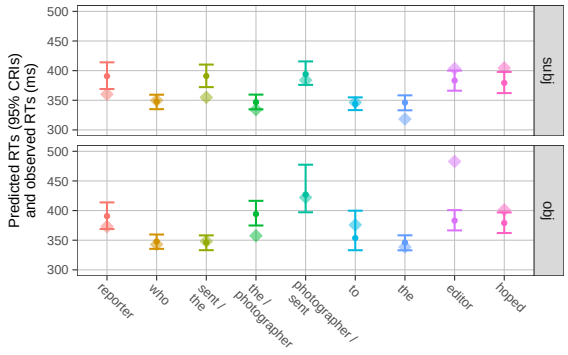


Figure 9: Model 3: 'parallel' reader

This increase in precision for Model 3 is clearly visible in its much lower $WAIC_2$ value:[3] Model 1 (postulated subject gaps) – $WAIC_1$ = 387.8, $WAIC_2$ = 1469; Model 2 (no postulated subject gaps) – $WAIC_1$ = 433.1, $WAIC_2$ = 1613; Model 3 ('parallel' reader) – $WAIC_1$ = 389.8, $WAIC_2$ = 553.4. These WAIC values provide a good summary of the conclusions we drew based on the posterior-prediction plots in Figures 7, 8 and 9: Model 1 is better than Model 2 with respect to both $WAIC_1$ and $WAIC_2$, and Model 3 provides sharper posterior predictions than either Model 1 or Model 2, as its $WAIC_2$ value shows (recall that $WAIC_2$ is variance based).

As a final way to evaluate our processing models, we can compare observed RTs and model predictions for individual words (in individual items) rather than focusing only on mean RTs, as we have done above. Let's focus only on the predictions made by Model 1; a linear regression with observed RTs for individual words as the response variable and predicted word RTs as the sole predictor estimates a slope of 1 (SE=0.009, t=5.7). That is, a 1 ms increase in predicted RTs corresponds to a 1 ms increase in observed RTs, indicating a very good data fit for our models at individual word level.

Finally, we used Model 1 to predict both eye-tracking (ET) and self-paced reading (SPR) data from Frank et al. (2013) (a variety of syntactic structures, no RCs; we selected 22 sentences that the parser, with its limited set of syntactic rules, parses correctly). For eye-tracking, we simply remove the key-press motor component from the model. Once again, we run a linear regression with observed and predicted word-level RTs as the response and predictor variables, respectively. The model fits both kinds of data fairly well: SPR – 1ms increase in predicted RT corresponds to 0.79ms increase in observed RT (t=2.1); ET – 1ms increase in predicted RT corresponds to 0.82ms increase in observed RT (t=3.31). The relative decrease in fit (the slope is not 1 anymore) is due to the fact that the model was really tailored to the RC data in Grodner and Gibson (2005, Exp. 1).

Conclusion

We introduced an extensible framework for mechanistic processing models and investigated 3 models incorporating an eager left-corner parser with visual and motor interfaces. The models differed in various theoretically-relevant respects, and the framework was used to quantitatively compare these different models / theoretical hypotheses.

We have only done quantitative comparisons based on posterior-prediction plots and WAIC values, but systematic across-the-board model comparison via Bayes factors is possible in the framework,

[3] Watanabe-Akaike/Widely Available Information Criterion; see Gelman et al. (2013, pp. 173-174) a.o. for discussion of both $WAIC_1$ and $WAIC_2$. We compute both WAIC values based on the estimated posterior `pyactr` RTs for the 18 ROIs (9 subject-gap ROIs + 9 object-gap ROIs).

as well as modeling a variety of other tasks (eye tracking, lexical decision).

Acknowledgments

We are grateful to Amanda Rysling, Donka Farkas, Abel Rodriguez, Matt Wagers, the UCSC S-lab audience (March 2018) and two ICCM 2018 anonymous reviewers for comments and discussion.

We want to thank Ted Gibson and Dan Grodner for providing the items and full datasets for the two experiments reported in their paper (Grodner and Gibson 2005). Jakub Dotlačil was supported by the NWO VENI 275-80-005 grant. The usual disclaimers apply.

References

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

Anderson, John R. and Christian Lebiere (1998). *The Atomic Components of Thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Brasoveanu, Adrian and Jakub Dotlačil (2018, in prep.). *Formal Linguistics and Cognitive Architecture: Integrating generative grammars, cognitive architectures and Bayesian methods*. Language, Cognition, and Mind (LCAM) Series. The pyactr library (Python3 ACT-R) is available here: https://github.com/jakdot/pyactr. Dordrecht: Springer.

Dillon, Brian et al. (2013). "Contrasting intrusion profiles for agreement and anaphora: Experimental and modeling evidence". In: *Journal of Memory and Language* 69.2, pp. 85–103.

Engelmann, Felix et al. (2013). "A Framework for Modeling the Interaction of Syntactic Processing and Eye Movement Control". In: *Topics in Cognitive Science* 5.3, pp. 452–474. DOI: 10.1111/tops.12026.

Frank, Stefan L et al. (2013). "Reading time data for evaluating broad-coverage models of English sentence processing". In: *Behavior Research Methods* 45.4, pp. 1182–1190.

Gelman, A. et al. (2013). *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis. ISBN: 9781439840955.

Grodner, Daniel and Edward Gibson (2005). "Consequences of the Serial Nature of Linguistic Input for Sentenial Complexity". In: *Cognitive Science* 29, pp. 261–291.

Hale, John T. (2014). *Automaton Theories of Human Sentence Comprehension*. Stanford: CSLI Publications.

Just, Marcel A. et al. (1982). "Paradigms and processes in reading comprehension". In: *Journal of Experimental Psychology: General* 111.2, pp. 228–238. DOI: 10.1037/0096-3445.111.2.228.

Kamp, Hans (1981). "A Theory of Truth and Semantic Representation". In: *Formal Methods in the Study of Language*. Ed. by Jeroen Groenendijk et al. Amsterdam: Mathematical Centre Tracts, pp. 277–322.

Kamp, Hans and Uwe Reyle (1993). *From Discourse to Logic. Introduction to Model theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Dordrecht: Kluwer.

Kieras, David E and David E Meyer (1996). "The EPIC architecture: Principles of operation". Unpublished manuscript from ftp://ftp. eecs. umich. edu/people/kieras/EPICarch. ps.

Kush, Dave W (2013). "Respecting relations: Memory access and antecedent retrieval in incremental sentence processing". PhD thesis. University of Maryland, College Park.

Lewandowsky, S. and S. Farrell (2010). *Computational Modeling in Cognition: Principles and Practice*. Thousand Oaks, CA, USA: SAGE Publications.

Lewis, Richard and Shravan Vasishth (2005). "An activation-based model of sentence processing as skilled memory retrieval". In: *Cognitive Science* 29, pp. 1–45.

Meyer, David E and David E Kieras (1997). "A computational theory of executive cognitive processes and multiple-task performance: Part I. Basic mechanisms." In: *Psychological review* 104.1, p. 3.

Murray, Wayne S and Kenneth I Forster (2004). "Serial mechanisms in lexical access: the rank hypothesis." In: *Psychological Review* 111.3, p. 721.

Nicenboim, Bruno and Shravan Vasishth (2018). "Models of retrieval in sentence comprehension: A computational evaluation using Bayesian hierarchical modeling". In: *Journal of Memory and Language* 99, pp. 1–34. DOI: https://doi.org/10.1016/j.jml.2017.08.004.

Resnik, Philip (1992). "Left-corner parsing and psychological plausibility". In: *Proceedings of the Fourteenth International Conference on Computational Linguistics*. Nantes, France.

Rij, Jacolien van (2012). *Pronoun processing: Computational, behavioral, and psychophysiological studies in children and adults*. Groningen.

Salvucci, Dario D (2001). "An integrated model of eye movements and visual encoding". In: *Cognitive Systems Research* 1.4, pp. 201–220.

Taatgen, Niels A and John R Anderson (2002). "Why do children learn to say "broke"? A model of learning the past tense without feedback". In: *Cognition* 86.2, pp. 123–155.

Vasishth, Shravan et al. (2008). "Processing Polarity: How the Ungrammatical Intrudes on the Grammatical". In: *Cognitive Science* 32, pp. 685–712.

Wagers, Matthew W et al. (2009). "Agreement attraction in comprehension: Representations and processes". In: *Journal of Memory and Language* 61.2, pp. 206–237.

Young, Richard M. and Richard L. Lewis (1999). "The Soar cognitive architecture and human working memory". In: ed. by Akira Miyake and Priti Shah, pp. 224–256.

Simulating Human-AI Collaboration with ACT-R and Project Malmo

Zachary M. Brill (zmb004@bucknell.edu), Christopher L. Dancy (christopher.dancy@bucknell.edu)
Department of Computer Science, Bucknell University
701 Moore Ave, Lewisburg, PA 17837 USA

Keywords: ACT-R; Project Malmo; teamwork; human-AI interaction; cognitive architectures, UCT.

Introduction

We use the ACT-R cognitive architecture (Anderson, 2007) to explore human-AI collaboration. Computational models of human and AI behavior, and their interaction, allow for more effective development of collaborative artificial intelligent agents. With these computational models and simulations, one may be better equipped to predict the situations in which certain classes of intelligent agents may be more suited to collaborate with people. One can more tractably understand and predict how different AI agents affect task behavior in these situations. To simulate human-AI collaboration, we are developing ACT-R models that work with more traditional AI agents to solve a task in Project Malmo (Johnson et al., 2016). We use existing AI agents that were originally developed as the AI portion of the Human-AI collaboration. In addition, creating a model in ACT-R to simulate human behavior gives us the opportunity to play out these interactions much faster than would be possible in real time.

Malmo Collaborative Challenge

The Collaborative AI Challenge was designed by Microsoft to test the collaborative capabilities of artificial intelligence. Built on top of Minecraft to create various environments and agents, Project Malmo is an attractive platform for experimenting with AI agents. Microsoft challenged teams to design and implement an artificial agent capable of playing alongside a human teammate in a game of Pig Chase, which is an extension of the Stag Hunt task developed by Yoshida, Dolan, and Friston (2008). This game requires a two-member team to track down and catch a pig within an enclosed meadow. Having only a limited number of actions (moving one square takes one action), cooperation is key to success. Cornering or "pinching" the pig with no escape route provides the maximum points to each agent for that round. However, the game also allows for a different kind of reward to be earned. If an agent gives up and exits the pen, they will also receive points, albeit fewer than if they had caught the pig.

The AI Agent

The primary AI agent, created by the Bacon Gulch team (Garriga, 2017), uses Bayesian inference and a planning algorithm to 1) determine the other player's approach and 2) plan its next move. The Bacon Gulch agent's Bayesian inference identifies the other player's strategy as either 'focused' or 'random'. The actions taken by the partner determine this inferred state. A 'focused' state is inferred when the teammate appears to share the same pig-catching goal as our artificial agent. If no clear movement towards the pig is being made, perhaps the teammate is standing in one spot spinning in circles, we assign their state to be 'random', and promptly exit the pig pen, acquiring a small reward. Using Bayes' theorem, we see the probability of action *a* being performed given the agent is in state *t* $P(a \mid t)$. With a strategy established, the AI agent must plan its next move, taking it closer to the goal state of capturing the pig.

UCT and Planning

The planning of moves is carried out by a domain-adapted variant of the Upper Confidence bounds for Trees (UCT) algorithm (Kocsis & Szepesvári, 2006). This variation on the Monte Carlo Tree Search (MCTS) algorithm shines in a simple task, such as the Pig Chase, where there is not sufficient complexity to require a learning algorithm (i.e., the playing field is small enough that all moves can be considered in a reasonable amount of time). The UCT algorithm builds a search tree that has as its root the initial state. This initial state is always the agent's current square. From this initial state, stochastic trajectories are simulated. A stochastic simulation is desired because it diminishes the detrimental effects of the pig's random movement on the agent's move planning, and allows for a better exploration of possible moves, since any valid move can be considered. A trajectory would be represented by a path in the tree, from which we could sum all the reward values along that path. This determines our expected outcome of following said trajectory. The path with the highest reward is chosen.

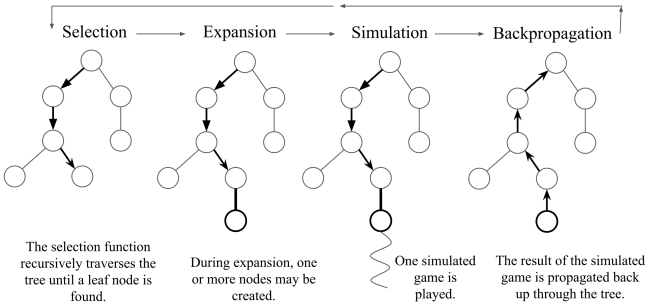


Figure 1: The Monte Carlo Tree Search algorithm procedure (based on Chaslot et al., 2008)

Figure 1 provides a general illustration of how the UCT algorithm works. Important to our research, is the simulation stage as previously mentioned. From this simulation, the final outcome, i.e. the score for the simulated round is backpropagated through the tree, allowing the agent to decide its best move given the current state.

Simulating the Human with ACT-R

We are developing a cognitive model of the human acting as part of a team in this task. We use these models to simulate and predict how differences in AI agents, and human cognitive states, may affect performance in human-AI collaboration. The declarative module houses all factual knowledge about the game's current state, primarily locations of all relevant objects e.g. 'me', my 'teammate', and the pig. ACT-R's procedural system (including utility) is used to complete actions. Lastly, we use ACT-R's motor module to make keystrokes, thus enabling the agent to play the game using perceptual-motor mechanisms similar to a human.

Model Strategy Overview

Previous work has shown that humans model their teammates as having similar decision-making processes to their own (Kennedy et al., 2008). In Kennedy et al.'s work, this projection of strategy improved the performance of a human-robot team. Similarly, our initial approach involves mirroring the AI agent's strategy within our model. This design choice will prove important in our analysis, enabling us to show shared strategies as a predictor of performance. This is the basis of our ACT-R model. Moreover, the model will grow to possess gameplay strategies that differ from the AI's, expanding our ability to analyze the effect of strategy similarity on the scores. Figure 2 details a high-level view of the model's approach to playing the game.

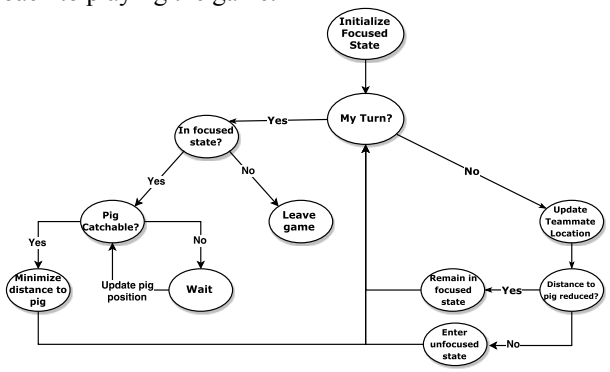


Figure 2: Initial model strategy. This strategy is designed to be as similar as possible to the AI's.

While many of the actual implementations within ACT-R are abstracted out of this flowchart, the visualization of this strategy shows the initial focused state and the ways in which the ACT-R agent not only makes its own moves, but how it responds to the moves of its teammate.

Foundations of the Model

The foundation of our model's strategy is an assumption of shared goals. To achieve the highest possible scores in the pig-chase game, both agents must attempt to catch or corner the pig. Thus, our model initializes itself with the assumption that its teammate will always attempt to reduce its distance to the pig. Our model identifies this as a focused state, the same state that is inferred by the AI agent. This state decision is implemented using a simple algorithm in which our agent pays attention to the teammate's movements. If, during the AI agent's turn, a negative move is made, i.e. their distance to the pig is increased, our agent determines them to be in an unfocused state, analogous to the AI agent's 'random' state. This unfocused state determination then shifts our agent's goal towards leaving the pen and earning itself a (small) reward. However, some adjustments to the model's goal change transition time would be valuable, as we could analyze the value of quitting and exiting immediately or waiting some amount of moves to allow the AI agent to recover a focused state.

Conclusion

The modelling of human behavior in a teamwork environment gives us the opportunity to not only add to previous research focused on human-machine interaction, but also contribute to our understanding of the aspects of human cognition that enable effective collaboration. Our primary goal is to observe the impact a shared strategy between teammates has on game performance. Using ACT-R and Project Malmo allows potential future expansion to more complex collaboration and environments, and the simulation of physiological and affective modulation of human-AI collaboration (e.g., using ACT-R/Φ, Dancy, 2013); both of these are important for the understanding of the consequences of integrating AI systems in different environments.

References

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

Chaslot, G. M. J. B., Winands, M. H. M., Uiterwijk, J. W. H. M., van der Herik, H. J., & Bouzy, B. (2008). Progressive strategies for monte-carlo tree search. *New Mathematics and Natural Computation, 4*(3), 343.

Dancy, C. L. (2013). ACT-RΦ: A cognitive architecture with physiology and affect. *Biologically Inspired Cognitive Architectures, 6*(1), 40–45.

Garriga, A. malmo-challenge, (2017), GitHub repository, https://github.com/rhaps0dy/malmo-challenge

Johnson, M., Hofmann, K., Hutton, T., & Bignell, D. (2016). The Malmo platform for artificial intelligence experimentation. *In proceedings of the Twenty-Fifth International joint conference on artificial intelligence (IJCAI)*, New York, NY, 4246-4247.

Kennedy, W. G., Bugajska, M. D., Adams, W., Schultz, A.C., & Trafton, J. G. (2008). Incorporating mental simulation for a more effective robotic teammate. *In proceedings of the Twenty-Third Association for the Advancement of Artificial Intelligence conference on artificial intelligence (AAAI),* Chicago, IL, 1300-1305.

Kocsis, L., & Szepesvári, C. (2006, 2006//). Bandit based monte-carlo planning. *In proceedings of the 17th European Conference on Machine Learning (ECML)*, Berlin, Heidelberg, 282-293.

Yoshida, W., Dolan, R. J., & Friston, K. J. (2008). Game theory of mind. *PLOS Computational Biology, 4*(12), e1000254.

Learning Effects Arise from Task-Indexed Adaptive Coding

S. Thomas Christie (tchristie@umn.edu)
Cognitive Science, 75 E River Rd
Minneapolis, MN, 55455 USA

Dominic Mussack (muss0080@umn.edu)
Department of Psychology, 75 E River Rd
Minneapolis, MN, 55455 USA

Paul R. Schrater (schrater@umn.edu)
Department of Psychology, 75 E River Rd
Minneapolis, MN, 55455 USA

Abstract

The brain is tasked with transmitting information quickly and reliably while limiting energy use. It must encode information without perfect knowledge of stimulus statistics, making efficient encoding impossible. We consider the possibility that the brain learns task-specific codes on-line, and that the efficiency of these codes can increase with exposure to task statistics. Use of task-indexed codes would parsimoniously explain, from information theoretic first principles, ubiquitous response time and accuracy improvements during task learning, including the Power Law of Learning. We also consider the implications of task-specific codes on automaticity and cognitive costs.

Keywords: learning; practice; information theory; decision making;

Introduction

Learning is characterized by adaptations of behavior which are typically assessed as increases in accuracy. During learning, however, reaction times often continue to improve after accuracy saturates (Shiffrin & Schneider, 1977). Learning theories based on Bayesian statistics make normative predictions about changes in accuracy (Jacobs & Kruschke, 2011), but do not explain continued improvements in timing after accuracy saturates. We propose that improvements in response time can be explained by learning to efficiently transmit state information. We thus make a novel proposal: learning involves *both* improving accuracy and finding efficient codes for transferring information within the brain.

Using tools from information theory to instantiate this assumption, we develop a normative theory that explains how the brain could learn a code with respect to a new task; this predicts the Power Law of Learning, an exponential-like decay in reaction time. To our knowledge, existing accounts of practice curves for response time and accuracy are either merely descriptive (see e.g. Newell & Rosenbloom 1981) or mechanistic (exploring implications of an algorithmic implementation, e.g. ACT-R (Anderson et al., 1997), the instance theory of automatization (Logan, 1988, 1992), or chunking (Miller, 1956)).

We start with a description of a problem the brain must solve: transferring information about the state of the world from sensory areas to action selection areas, i.e. across an information channel. In order to transmit this information

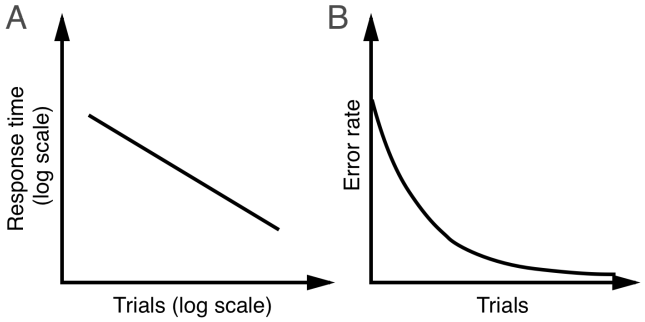


Figure 1: (A) The Power Law of Learning (Newell & Rosenbloom, 1981) describes an exponential decrease in response times as a function of practice, which appears linear on log-log scale. (B) Error rates show exponential-like decreases with practice in many tasks (Heitz, 2014).

quickly, the brain should construct an efficient code of state information, which in turn depends on stimulus statistics and must be learned. When encountering a new task, subjects must learn many things, including which sensory information in the environment is relevant to the task (feature extraction) and which actions provide the highest reward rate over time (policy learning) as a function of time-varying circumstances (state representation). Efficiently transferring and updating state information is a problem distinct from these. In the current paper, we introduce an ideal-observer model of the efficient transmission problem, and show that learning an efficient encoding gives rise to curves that qualitatively match practice curves seen in the literature (see Figure 1).

The current investigation can be seen as an application of the Efficient Coding Hypothesis (ECH) to the time scale of task learning. First proposed in the 1960s, ECH suggests that the brain develops codes designed to transmit the maximum amount of information with the fewest neural spikes (Barlow, 1961). This hypothesis has been supported by evidence from the visual system of the fly (Laughlin, 1981), the primate visual area V1 (Vinje & Gallant, 2000), and the retina of macaques and salamanders (Pitkow & Meister, 2012). In each case, an efficient coding scheme is constructed using knowl-

edge of stimulus statistics, which are learned on evolutionary or developmental time scales (Simoncelli & Olshausen, 2001). We propose that an analogous situation pertains to task-related information learned on-line during task engagement.

In what follows, we introduce our proposal using relevant concepts from information and control theory. We include a simple worked example with numerical simulations to demonstrate the production of response time and accuracy curves as a function of coding efficiency. We then show that these curves arise from Binomial, Multinomial, and Gaussian stimulus distributions. Finally, we discuss the relationship between the current work and adaptive codes in the visual system, as well as implications for understanding automaticity and cognitive costs.

Efficient state encoding

When behaving on-line in an ecological environment, an organism must represent environmental information sufficient to afford action selection. We assume that this information must be transmitted from the senses, through the brain, which will eventually result in an action. In this context, the brain acts as an information channel, transmitting information between sensory input and action output.

While our approach is reminiscent of applications of information theory to psychology from the 1950s and later (see Lachman et al. 1979 for a review), we explicitly restrict ourselves to David Marr's computational level of analysis (Marr, 1982), in which we investigate the problem the brain is trying to solve: transmitting stimulus information quickly over a finite-capacity channel.

Problem description We borrow from control theory and say that information about the world is compressed into a state s, where the space of possible states S and the world-information-to-state mapping are determined by the task being performed. Once the value of the state s is computed (say, for a single trial of an experiment), it must be transmitted to an action selection mechanism. This transmission process is the concern of this work. In order to be transmitted over a channel, s must be encoded into a form transmissible by the channel medium (in this case, the brain). In accordance with the computational level of analysis, we make no claim about the form of the code, except to point out that that the same information can be transmitted using either efficient or inefficient codes. Efficient codes are preferable from the perspective of maximizing transmission rates and reducing transmission costs.

We treat the state as drawn from a probability distribution $P(S)$. In general, the quantity of information that must be transmitted in order to specify the state is equivalent to the entropy of the state space:

$$H(S) = -\sum_{s \in S} P(S=s) \log_2 P(S=s)$$

In an efficient code, the average code length assigned to each value of s is proportional to $-\log_2 P(S=s)$. As such, constructing an efficient code requires knowledge of the distribution of $P(S)$ (Cover & Thomas, 2012).

Inefficient code length Critically, $P(S)$ must be learned from experience, as individuals engaged in a new task do not know the state frequency distribution. An inefficient coding of a state s will result in a longer transmission than an efficient code. Assume that an inefficient code is constructed using an estimate of the state space distribution $Q(S)$. The extra code length is defined by the Kullback-Leibler divergence between the two distributions, usually written as D_{KL} or KL:

$$KL(P||Q) = \sum_{s \in S} P(S=s) \log_2 \frac{P(S=s)}{Q(S=s)}$$

As an individual performing a task observes more realizations of the stimulus state, they can update Q, and with many observations $KL(P||Q) \to 0$. This in turn allows the subject to construct an (approximately) efficient code, transmit state information in a minimum amount of time, and select an action more quickly. Thus, reaction times directly depend on the efficiency of state encoding.

A simple example Suppose an individual is engaged in a task in which they observe the outcome of a coin flip. They are instructed to press a left arrow key if they observe heads, and a right arrow key if they observe tails. The image of a coin must be processed visually and turned into a state $s \in S$ (object recognition or feature extraction), and the mapping between flip outcome and response is learned (policy learning). These processes contribute to response accuracy. In addition, the state of each flip must be encoded and transmitted through the brain to an action selection mechanism that in turn drives a key press. This second process contributes to response time.

Suppose the probability of a biased coin landing on heads was $P(S = \text{heads}) = 0.9$, and each flip is modeled as being drawn from a Bernoulli distribution. For an unbiased coin, the stimulus encoder mechanism (sender) could transmit $\log_2 2 = 1$ bit of information to the action selector mechanism (receiver). If the sender knew the bias of this particular coin, it would only need to transmit $H(S) = -(0.9 \log_2 0.9 + 0.1 \log_2 0.1) = 0.47$ bits, on average, to specify the outcome of each coin flip.

Assume that the sender has a prior belief that the coin is unbiased, and thus has an estimate Q_{init} of P. This subjective ignorance means that the sender is unable to construct an optimal code. Hence, the sender would construct a code that requires 0.53 extra bits:

$$\begin{aligned} KL(P||Q_{init}) &= \sum_{s \in S} P(S=s) \log_2 \frac{P(S=s)}{Q(S=s)} \\ &= 0.9 \log_2 \frac{0.9}{0.5} + 0.1 \log_2 \frac{0.1}{0.5} \\ &= 0.53 \end{aligned}$$

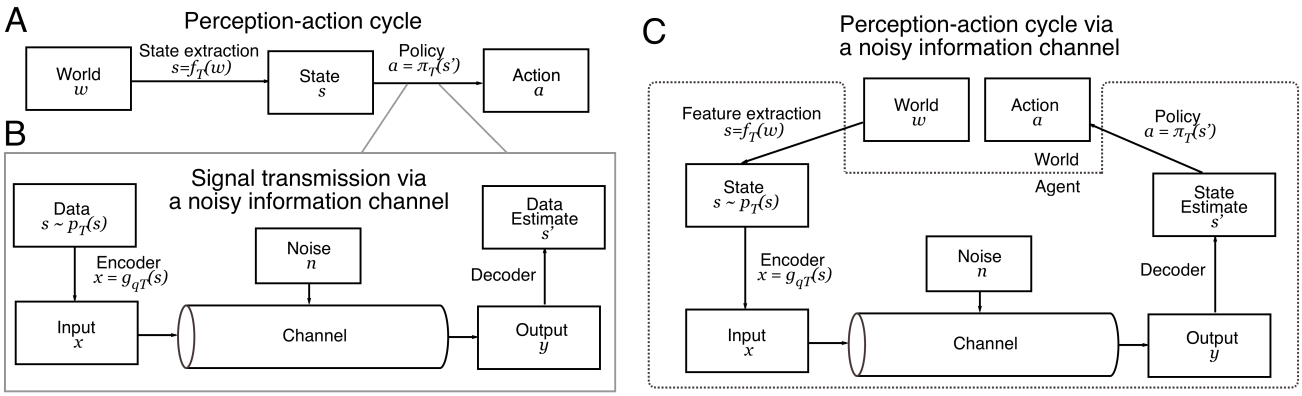


Figure 2: (A) Typical decision making models involve transforming observations into a state, and then using the state to select an action. (B) We claim that the transmission of state information to the action selection mechanism is subject to the constraints of information transmission: the state must be encoded and transmitted through a noisy channel. This figure is adapted from Stone (2015), with permission. (C) Taken together, we propose a refined model of action selection that accounts for transmission time, which provides a normative explanation for practice effects. State information s is encoded using a task-specific encoding (g_{qT} for a task T) using an approximation q of the state frequency distribution p. The efficiency of this encoding depends on the quality of the estimate q.

Power Law of Learning

As the sender observes more coin flips, its estimate Q_{init} of P improves to Q_{obs}, and the sender's code becomes more efficient. To investigate the characteristics of this update, we simulated 10,000 coin flips for 2,000 subjects. For each flip we calculated the maximum likelihood estimate Q_{obs} of P using every observed flip to that point, then calculated $KL(P||Q_{obs})$. We then averaged the KL-divergence across all subjects for each number of flips. Results are shown in Figure 3. When subjects have a weak prior, the code length improvement has a linear relationship with stimulus observations when plotted in log-log space, exactly like the Power Law of Learning (see Newell & Rosenbloom 1981). However, the initial rate of average code length decrease depends on the strength of subjects' prior beliefs that the coin is unbiased. Code length improvement rates as a function of prior strengths are shown in Figure 3C.

In our simulations, the linear power law relationship is clearest when our virtual subjects have weak priors on stimulus statistics. Accordingly, we suspect that the Power Law of Learning arises as an experimental result exactly because this is akin to the scenario in many psychology experiments, in which subjects are unfamiliar with the task and have a weak prior on stimulus statistics. The existence of strong, incorrect prior beliefs results in non-exponential learning curves, an experimentally testable prediction.

Generality In the coin flip example, we focus on encoding the value of draws from a Bernoulli distribution and show power law decreases in transmission time. In Figure 3D, we show similar results for both Multinomial ($k = 16$) and univariate Gaussian distributions. These simulation results suggest that the trend may not depend on the specific parameters or form of the state distribution. Interestingly, each case results in a power law exponent of -1. Power law fits to response time curves in the experimental literature give exponents between 0 and -1 (Newell & Rosenbloom 1981), indicating a slower rate of learning in real subjects than the ideal observer model presented here.

Aggregating across subjects In the psychology literature, the smooth linear curves described by the Power Law of Learning arise as a result of averaging across many subjects. This observation has been leveled against Power Law of Learning as a criticism (Heathcote et al., 2000), but as we show below, this power-law decay is an effect of aggregating many subject response times when each subject is learning optimally. Indeed, an organism using efficient coding should have trial-to-trial response time variance related to both their updated code and the information content ('surprise') of each observed state. Therefore, each subjects' trend during learning should not be smooth. The between-subject variance of simulated response times is shown as vertical gray lines in Figure 3B.

Error rate improvements

As encoding efficiency increases with task practice, more information can be transmitted in a given unit of time. Forcing a rapid response is known to lead to higher error rates in many tasks (Heitz, 2014), presumably because subjects have less time to incorporate information from their environment. In the model we are proposing, a subject observes a stimulus and must still transmit the value of that stimulus through the brain to an action-selection mechanism. If a subject had unlimited time to respond, they could send the value of the stimulus with perfect fidelity. However, forcing a rapid response introduces a scenario in which the sender may only

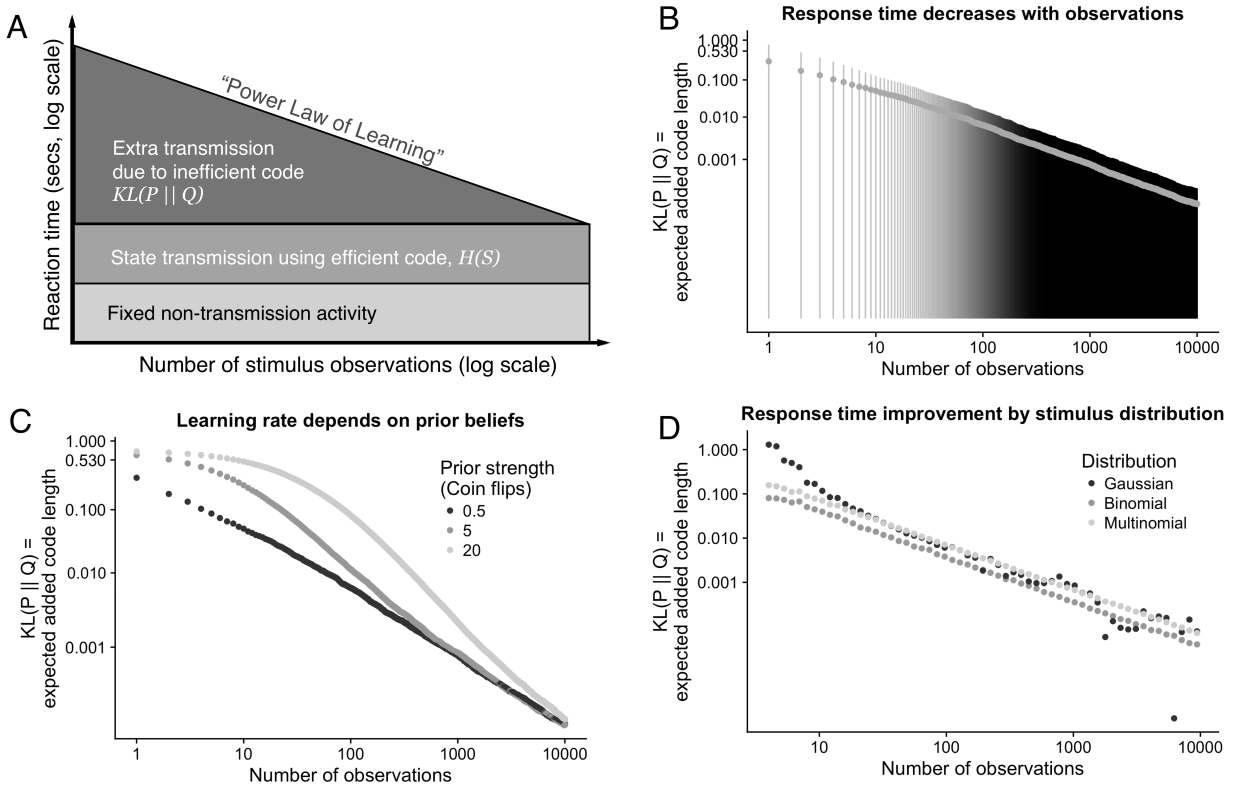


Figure 3: (A) Response time is composed of fixed time costs (including the cost of efficiently transmitting state information), plus extra transmission time due to use of an inefficient code, which decreases with practice. (B) This figure shows change in $KL(P||Q)$ as flips of a biased coin are observed, averaged across many subjects. $KL(P||Q)$ measures the inefficiency of state encoding, and decreases linearly in log-log space. This mirrors the Power Law of Learning. Grey lines represent 1 SE of non-transformed values. (C) A strong, incorrect prior on the state frequency distribution initially slows learning. Each line represents a different strength prior belief that an biased coin is actually unbiased. (D) The power-law relationship holds for observations of state spaces following Binomial, Multinomial, and Gaussian distributions. In each simulation, values are averaged across 2000 simulated subjects.

have time to transmit a partial representation of the stimulus.

Each transmission results in a corresponding reduction in the uncertainty about the state from the perspective of the action selection mechanism. A limitation on transmission time (such as a forced-response time, or a task that rewards speed over accuracy) could result in incomplete transmission of the state, an incomplete reduction in the receiver's uncertainty, and a corresponding error in action selection. Error rates due to incomplete task transmission should decrease as encoding efficiency increases.

Suppose that in our coin flip example, the stimulus information channel could sustain transmissions of 1 bit/second. Suppose further that we, as experimenters, force a 500ms response time. For the sake of this example, we assume that non-transmission-related activities take zero time. The sender's initial code based on an assumption of coin fairness Q_{init} requires 1 bit to transmit the flip outcome. Only 0.5 bits can be transmitted before a response is required. We should expect the sender to select actions with some non-zero error rate.

We can calculate the expected error rate by recalling that the entropy of the coin flip $H(S)$ is from the perspective of the *receiver*. The entropy equation $H(S) = -\sum_{s\in S} Pr(S = s)\log_2 Pr(S = s)$ provides a way of converting the remaining uncertainty into a probability.

$$\begin{aligned} H_{remaining}(S) =& 0.5 \\ =& -\sum_{s\in S} Pr(S=s)\log_2 Pr(S=s) \\ =& -Q_{init}(S=H)\log_2(Q_{init}(S=H)) - \\ & (1-Q_{init}(S=H))\log_2(1-Q_{init}(S=H)) \end{aligned}$$

Solving for $Q_{init}(S=H)$ gives 0.89, or an 89% chance that the partial message received actually indicates heads. As more samples are observed and the code is improved, eventually a code is constructed with which only 0.47 bits must be transmitted, as shown above. Using this code, the outcome of the coin flip can be transmitted over the channel in $<$ 500ms.

We calculated expected error rates for transmitting the value of the biased coin as a function of observations, and therefore coding efficiency, for several forced-choice times. We show the results in Figure 4.

Speed-accuracy tradeoff When subjects respond more quickly, they often respond with lower accuracy (Heitz, 2014). This relationship is known as the Speed Accuracy Tradeoff (SAT). The SAT is typically studied using stochastic, dynamic stimuli such as the moving dot stimulus (see e.g. Ball & Sekuler 1982). In these cases, subjects accumulate information about the stimulus state over time. If forced to respond quickly, or if given little time to accumulate information, their accuracy suffers.

Recognizing that the transmission of state information in the brain is not instantaneous - the position we take in this paper - affords the possibility of investigating the SAT with static stimuli. As noted above, a subject observing a stimulus that requires 1 bit of information to be transmitted over a 1 bit/second channel, but given only 500ms to transmit the information, should suffer a decrease in accuracy. Varying the entropy of the stimulus state space and the time subjects have to respond should result in speed-accuracy curves like those shown in Figure 4C. This is another testable prediction of our theory.

Task-indexed adaptive encoding We emphasize that, as with action selection policy and state space definition, state encoding is determined with respect to the task being performed. When a person encounters a new task with a new state space, forming a new code specific to the task will enable efficient transmission in the long run. We call task-specific code building *adaptive* because it involves adapting a neural code to a specific task.

Discussion and conclusion

In this paper we introduce a normative theory of task-specific efficient code learning. This theory provides a generative explanation for the Power Law of Learning and exponential-like decay curves without specifying a particular decision model. Our only requirement on the decision process is that the state information for a task be transmitted through a finite-capacity channel. As such, this theory can be integrated into other decision making frameworks, in particular neural or dual-processing theories.

Neural plausibility The Efficient Coding Hypothesis describes a principle for optimal information transmission over the lifespan of an organism interacting with their environment; adaptation occurs on the evolutionary or developmental timescale. Recent evidence suggests that neural codes can also adapt at the millisecond timescale, apparently in order to maintain a high degree of information transmission in the face of rapidly changing stimulus statistics (Fairhall et al., 2001; Brenner et al., 2000). These findings suggest that the Efficient Coding Hypothesis may also apply to a fast temporal scale in addition to the relatively static tuning curves and

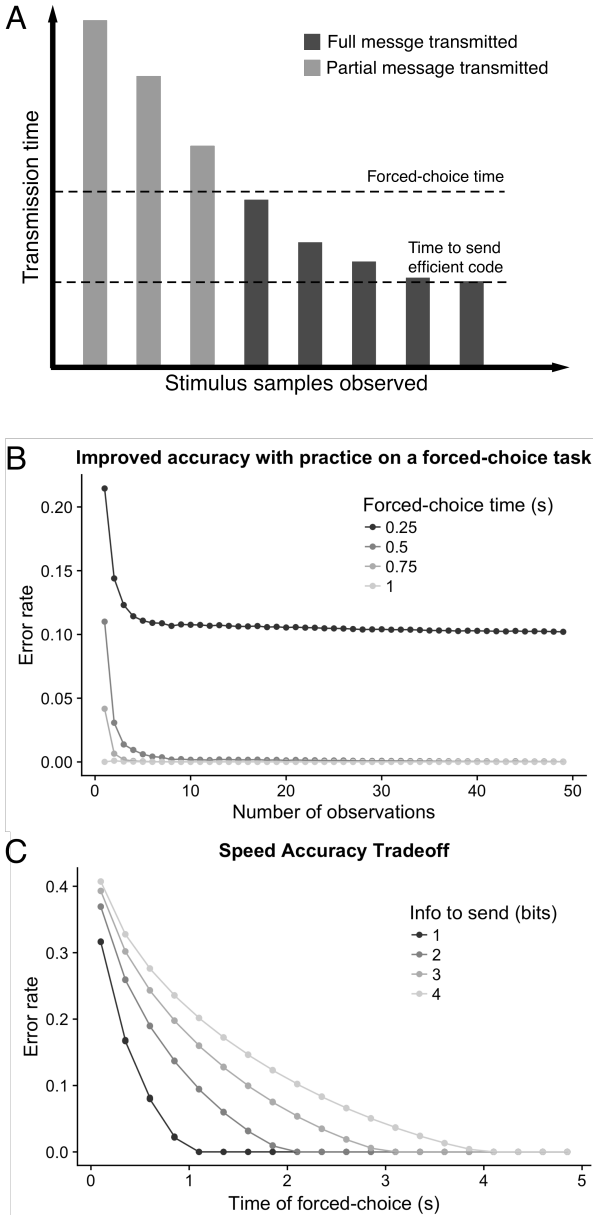


Figure 4: (A) Each bar represents the transmission of the stimulus state on a trial. Limiting transmission time by forcing a response can result in incomplete transmission of the state. Nevertheless, more information can be transmitted as coding efficiency improves. Zero error can only be achieved when encoding is efficient enough to allow complete transmission of the state information. (B) The outcome of a biased coin flip takes 470ms to transmit over a channel at a rate of 1 bit/s. An initially inefficient code may result in incomplete state transmission depending on the response time allowed, resulting in errors. Errors decrease as encoding improves. (C) Varying forced-choice time results in speed-accuracy tradeoff curves due to the resulting variable transmission of state information.

cell sensitivities that are the typical focus of analysis. In our view, such findings support the plausibility of task-specific code adaptation.

Relation to automaticity Work on automaticity and implicit learning emphasizes that attentional capacity, response time, accuracy, and multitasking performance all increase with practice and the development of expertise in a task (Logan, 1988). However, these improvements appear to be strongly tied to specific tasks and environments. Learning task-specific coding suggests a normative account of these effects. When transmitting information over a limited-capacity channel, on-line learning of an efficient code enables more information to be transmitted in a given time. We suggest that task automaticity may be equivalent to learning an efficient task-specific code.

Cognitive costs Cognitive operations are referred to as 'costly' when they give rise to subjective fatigue and behavioral aversion (see Shenhav et al. 2017 for a recent review). We note that task aversion often subsides with practice and the development of task-specific expertise, which appears related to the development of task automaticity. For that reason, we speculate that decrease in cognitive costs with practice may be a direct result of learning an efficient encoding of task states, and conversely that the the costs themselves arise from transmitting information using inefficient codes. Minimizing transmission costs may also reduce metabolic costs, which have been proposed as a primary driver of cognitive costs (Christie & Schrater 2015).

Limitations and future work We have proposed that the brain learns efficient codes on-line, but have intentionally avoided specifying algorithmic or neural implementation details. In order to model behavioral data from a specific task, we would need to specify a full decision model, an encoding algorithm, and specify the characteristics of the information channel, e.g. channel noise. Our current analysis serves as a normative ideal-observer model against which actual behavioral data can be compared.

References

Anderson, J. R., Matessa, M., & Lebiere, C. (1997). Act-r: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction*, *12*(4), 439–462.

Ball, K., & Sekuler, R. (1982). A specific and enduring improvement in visual motion discrimination. *Science*, *218*(4573), 697–698.

Barlow, H. B. (1961). Possible principles underlying the transformations of sensory messages.

Brenner, N., Bialek, W., & Van Steveninck, R. d. R. (2000). Adaptive rescaling maximizes information transmission. *Neuron*, *26*(3), 695–702.

Christie, S. T., & Schrater, P. (2015). Cognitive cost as dynamic allocation of energetic resources. *Frontiers in neuroscience*, *9*, 289.

Cover, T. M., & Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Fairhall, A. L., Lewen, G. D., Bialek, W., & Steveninck, R. R. d. R. van. (2001). Efficiency and ambiguity in an adaptive neural code. *Nature*, *412*(6849), 787.

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

Heitz, R. P. (2014). The speed-accuracy tradeoff: history, physiology, methodology, and behavior. *Frontiers in neuroscience*, *8*, 150.

Jacobs, R. A., & Kruschke, J. K. (2011). Bayesian learning theory applied to human cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, *2*(1), 8–21.

Lachman, R., Lachman, J. L., & Butterfield, E. C. (1979). *Cognitive psychology and information processing. hillsdasle.* Lawrence Erlbaum.

Laughlin, S. (1981). A simple coding procedure enhances a neuron's information capacity. *Zeitschrift für Naturforschung c*, *36*(9-10), 910–912.

Logan, G. D. (1988). Toward an instance theory of automatization. *Psychological review*, *95*(4), 492.

Logan, G. D. (1992). Shapes of reaction-time distributions and shapes of learning curves: A test of the instance theory of automaticity. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *18*(5), 883.

Marr, D. (1982). Vision: a computational investigation into the human representation and processing of visual information. w. h. *WH San Francisco: Freeman and Company*.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, *63*(2), 81.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. *Cognitive skills and their acquisition*, *1*(1981), 1–55.

Pitkow, X., & Meister, M. (2012). Decorrelation and efficient coding by retinal ganglion cells. *Nature neuroscience*, *15*(4), 628.

Shenhav, A., Musslick, S., Lieder, F., Kool, W., Griffiths, T. L., Cohen, J. D., et al. (2017). Toward a rational and mechanistic account of mental effort. *Annual review of neuroscience*, *40*, 99–124.

Shiffrin, R. M., & Schneider, W. (1977). Controlled and automatic human information processing: Ii. perceptual learning, automatic attending and a general theory. *Psychological review*, *84*(2), 127.

Simoncelli, E. P., & Olshausen, B. A. (2001). Natural image statistics and neural representation. *Annual review of neuroscience*, *24*(1), 1193–1216.

Stone, J. V. (2015). *Information theory: A tutorial introduction*. Sebtel Press.

Vinje, W. E., & Gallant, J. L. (2000). Sparse coding and decorrelation in primary visual cortex during natural vision. *Science*, *287*(5456), 1273–1276.

Models of Bayesian Rationality for Conditional Reasoning: What are they good for?

Lukas Elflein and **Marco Ragni** {**elfleinl, ragni**}**@cs.uni-freiburg.de**
Cognitive Computation Lab, Georges-Koehler-Allee 79, University of Freiburg, 79111 Freiburg, Germany

Problem Description

A conditional is a statement that can be formulated by using the word 'if', e.g., 'if a disease is contagious then the illness can spread'. It is often used for scientific reasoning, causal reasoning or expressing relations between a precondition and a consequent. If additional information is given e.g., a disease is contagious or an illness cannot spread then inferences can be drawn. We write 'if p then q' for the conditional. There are four classical reasoning types: Modus Ponens (MP), Modus Tollens (MT), Affirming the Consequent (AC), and Denying the Antecedent (DA), we write for the inference (short: $\therefore$):

$$MP: \frac{p \rightarrow q, p}{\therefore q} \qquad MT: \frac{p \rightarrow q, \neg q}{\therefore \neg p} \tag{1}$$

For abstract conditionals, Modus Ponens is accepted by most people (97%), while Modus Tollens is accepted far fewer participants (57%) (Oberauer, 2006). AC and DA are invalid but are still accepted fairly often (44% for AC and 38% for DA):

$$DA: \frac{p \rightarrow q, \neg p}{\therefore \neg q} \qquad AC: \frac{p \rightarrow q, q}{\therefore p} \tag{2}$$

The theory of Bayesian Rationality (Oaksford, Chater, & Larkin, 2000) claims that human reasoning about a conditional 'if p, then q', can be described by the conditional probability $P(q|p)$. The most prominent approach is the model by Oaksford et al. (2000). The models for the Wason selection task, i.e. the Dependence/Independence Models (Oaksford & Chater, 1994) realize a similar probabilistic approach that we also analyze. The three theories contain up to 3 fitting parameters each, with analogous interpretations: Where $P(p)$ is the probability that 'p' is plausible in the real world, analogous for 'q'. $P(\neg q|p)$ can be seen as the probability that the stated 'if... then' rule is implausible. Now that we have introduced the models of Bayesian Rationality, the main question remains: What do they model? There are (at least) two intuitive interpretations of the Bayesian Rationality approach.

Table 1: Bayesian Rationality models for acceptance of conditionals with $a = P(p)$, $b = P(q)$, $e = P(\neg q|p)$

	Oaksford 2000	Dependence	Independence
MP	$1-e$	1	b
MT	$\frac{1-b-a\ e}{1-b}$	$1-a$	$1-a$
AC	$\frac{a\ (1-e)}{b}$	$\frac{a}{b}$	a
DA	$\frac{1-b-a \cdot e}{1-a}$	$1-b$	$1-b$

1. Individual level: Each reasoner employs similar cognitive/information processes. Bayesian Rationality describes these general processes. They can be described by one set of point-estimates of parameters that describes the reasoners.
2. Group level: In a group, reasoning processes may differ between individuals, but the parameter point estimates are a good description of the distribution of parameters.

We now put these two interpretations to the test.

Experimental Data from Oberauer (2006)

Often experimental reports in the literature just focused on the percentages of participants endorsing each of the individual patterns MP, MT, AC, and DA. A study conducted by Oberauer (2006) instead reports answer patterns of individual participants. These are shown in Fig. 1 for two different abstract conditionals (card-number and triangle-square). We will abbreviate each reasoners answer by the tuple (MP, MT, AC, DA). Many reasoners seem to accept all four inference patterns or some few combinations (e.g., 'only MP', which we write as '1000'). Most other patterns (e.g., 'only DA', '0001') were observed almost never. This already hints at a substantial inhomogeneity in the responses of the human reasoners in reasoning about conditionals. This diversity

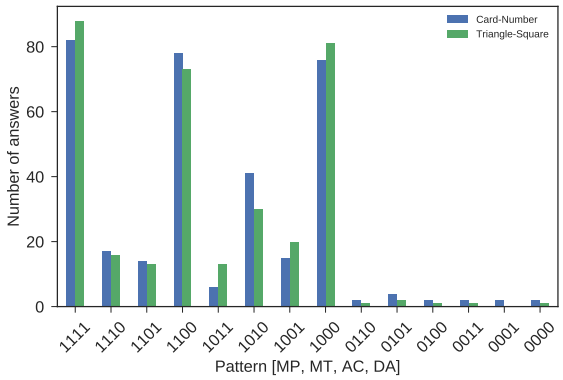


Figure 1: The basic inference patterns taken from Oberauer (2006) study on conditionals. '1' is acceptance and '0' is a rejection of the modus.

gets more pronounced if we do not pool the answers independently, but analyze how subjects answered jointly for two different conditionals (Fig. 2). Approximately 50% of the participants are internally consistent in their answer patterns, all others accept different conditionals when changing from the card-number condition to the triangle-square condition.

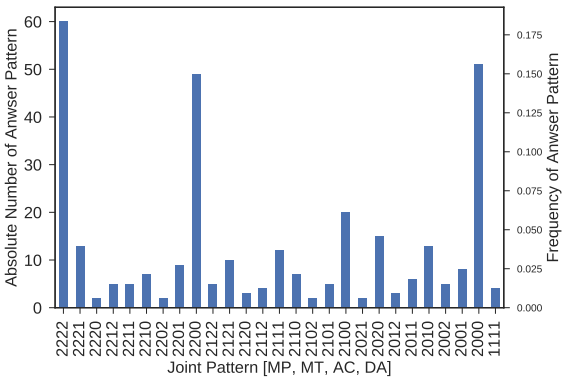


Figure 2: Joint inference patterns from Oberauer (2006), excluding patterns which appear only once. '2' means that the conditional modus was accepted in both experimental conditions.

Fitting the Models & Conclusion

How does this inhomogeneity in the data impact the model parameters? We obtain best-fit estimates by minimizing the Root Mean Squared Error between model prediction and experimental data using the L-BFGS-B algorithm from the SciPy package in Python. For the models presented in Table 1, best-fit parameters can be estimated in two ways: On the one hand, individual data aggregated into the four categories MP, MT, AC, DA can be used to obtain a single parameter value per experiment. This point estimate is shown in

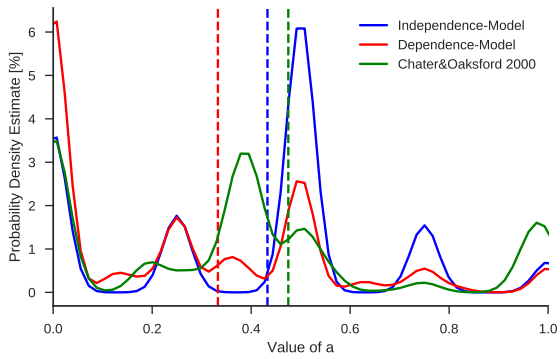


Figure 3: Estimated distribution of the parameter $a = P(p)$ show multimodal distributions (curves) for three models. Some of the aggregate point estimates agree roughly with the central mode (red and blue dashed lines), some do not (green)

Fig. 3 and Fig. 4 as a dashed vertical line. On the other hand, the full distribution of individual best-fit parameters can be estimated. Each point estimate is weighted by its relative frequency in the population of subjects in the experiment. The distribution is smoothed via a kernel density estimation (gaussian, bandwith 0.1). In Fig. 3 we can see the distribution of the 'a' parameter, which reflects the probability of the antecedent being true. The distribution has multiple local maxima (or modes).The aggregate point estimates are close to the central mode at 0.5. The 'b' parameter, which reflects the probability of the consequent being true, is also distributed multimodally, with local maxima around 0, 0.5 and 1. The aggregate point estimates are closer to 'True' (0.7).

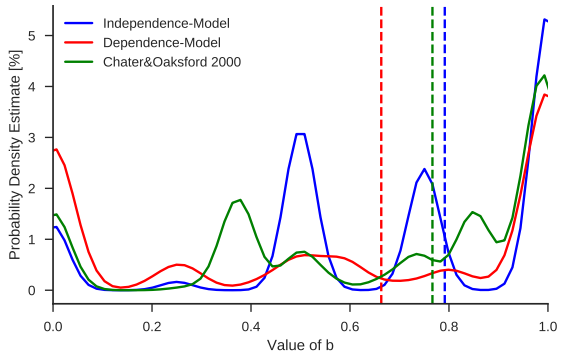


Figure 4: Multimodal distribution of the kernel density estimates of the parameter $b = P(q)$ (curves). Most aggregate point estimates (dashed vertical lines) are far from the global maxima, some are close to local minima.

The *first interpretation*, i.e., 'the models describe one universal cognitive process' is incompatible with the parameter fits. If there was only one process that all humans used which was described by the model parameters, these parameters should be distributed unimodally – e.g., in a Gaussian distribution with some (preferably small) dispersion. The *second interpretation*, i.e., 'the models describe how the majority of the group reasons', is also not valid. If this was the case, parameters estimated from aggregated data would coincide with (global) maxima of the individual parameter distributions. For the 'b' parameter (Fig. 4), this is clearly not the case. Thus, the models do not describe the behavior of the majority of the group in a meaningful way.

In future research, we will adapt these models to capture processes and predict answers of individual reasoners.

Acknowledgments

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

References

Oaksford, M., & Chater, N. (1994). A rational analysis of the selection task as optimal data selection. *Psychological Review*, *101*(4), 608.

Oaksford, M., Chater, N., & Larkin, J. (2000). Probabilities and polarity biases in conditional inference. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *26*(4), 883.

Oberauer, K. (2006). Reasoning with conditionals: A test of formal models of four theories. *Cognitive Psychology*, *53*(3), 238–283.

An Integrated Working Memory Model For Time-Based Resource-Sharing

Joseph J. Glavan (glavan.3@wright.edu)
Joseph W. Houpt (joseph.houpt@wright.edu)
Department of Psychology, Wright State University
3640 Colonel Glenn Hwy. Dayton, OH 45435 USA

Abstract

The time-based resource-sharing (TBRS) model envisions working memory as a rapidly switching, serial, attentional refreshing mechanism. Executive attention trades its time between rebuilding decaying memory traces and processing extraneous activity. To thoroughly investigate the implications of the TBRS theory, we integrated TBRS within the ACT-R cognitive architecture, which allowed us to test the TBRS model against both participant accuracy and response time data in a dual task environment. In the current work, we extend the model to include articulatory rehearsal, which has been argued in the literature to be a separate mechanism from attentional refreshing. Additionally, we use the model to predict performance under a larger range of cognitive load than typically administered to human subjects. Our simulations support the hypothesis that working memory capacity is a linear function of cognitive load and suggest that this effect is less pronounced when articulatory rehearsal is available.

Keywords: working memory; time-based resource-sharing; ACT-R; attentional refreshing; articulatory rehearsal; computational modeling

Introduction

Working memory (WM) is vital for elemental cognitive processing. It provides the cognitive system with the means for manipulating and retaining information, such as the intermediate solution to an algebra problem, for short periods of time. WM is often thought of as a storage system with a structurally limited capacity (Case, Kurland, & Goldberg, 1982), similar to a physical container. However, evidence is growing for the interpretation of WM as a mechanism whose active maintenance yields a functional capacity (Barrouillet & Camos, 2015). That is to say that the actions of the WM mechanism cause more items to remain accessible than it can strictly store at a single time (e.g., in a buffer).

The time-based resource-sharing (TBRS) model (Barrouillet, Bernardin, & Camos, 2004; Barrouillet & Camos, 2015) advocates this view, proposing that WM is a rapidly switching, serial mechanism that focuses executive attention on a single memory trace at a time to rebuild its activation (i.e., accessibility). This process, called attentional refreshing, counteracts the continuous temporal decay experienced by items in memory. While all memory items decay, only one item can be refreshed at a time because of a central bottleneck. To maintain access to a set of items, the items must share their time (i.e. take turns) in the focus of attention.

A popular paradigm for studying WM is a dual task where the subject must memorize a list of items while responding to some distractor task. The distractor task is intended to prevent the use of higher-level strategies (i.e. mnemonics) so that the number of items recalled, *span*, reflects a more pure WM capacity. Barrouillet et al. (2004) recognized that this approach may reduce the subject's ability to engage in target maintenance, but it cannot completely eliminate it. Instead, they sought to carefully control and quantify the interference from the distractor activity as cognitive load (CL).

In support of TBRS, Barrouillet, Camos, and colleagues (e.g., Barrouillet et al., 2004; Barrouillet, Bernardin, Portrat, Vergauwe, & Camos, 2007) have repeatedly demonstrated that the essence of CL is time away from maintenance, rather than the number of distractors or their inherent difficulty. By using distractor task response times as a proxy for distractor processing time, they demonstrated that observed WM span is an approximately linear function of CL. Specifically, if CL is measured as the fraction of the total time on task dedicated to the distractor task, then span $= k(1-\mathrm{CL})$. In this formulation, k is a free parameter reflecting an individual's "pure" WM capacity. A meta-analysis of 14 experimental conditions indicated excellent fit for this function ($R^2 = .98$) and close agreement with Miller's Magical Number 7 ± 2: span $= -8.33\mathrm{CL} + 8.13$ (Barrouillet, Portrat, & Camos, 2011).

One of the goals of this paper is to explore whether the linear relationship between CL and span holds for more extreme values. The mean CLs of the experimental conditions from which the preceding regression comes range from approximately .25 to .65. By using measurements from the middle of the CL continuum, it is possible that we have only observed the locally linear, middle section of a function that is actually non-linear (e.g., ogival). In other words, testing WM span at the extremes of CL would provide a stronger test of the linear function hypothesis. We explore this possibility by using our model to simulate the full range of CL.

Oberauer and Lewandowsky (2011) took a similar approach with their connectionist model, TBRS*, and simulated a much more complete range of the CL continuum. While their model did find support for a linear effect of CL on WM span, it makes a simplifying assumption that may limit its correspondence with TBRS; TBRS* abstracts away the distractor processing portion of the task. To simulate each episode of cognitive load, it samples a response time from a distribution determined post-hoc from empirical observations. The critical insight of TBRS is that observed WM capacity is a product of the interaction of maintenance and processing; therefore, a computational model of TBRS must be integrated to explain both the memory and processing tasks. As we will see, nuances in distractor processing can have remarkable effects on WM span.

The second goal of our paper is to integrate articulatory re-

hearsal into the computational TBRS model. While the original TBRS model regarded attentional refreshing as the primary mechanism behind memory maintenance, more recent work has highlighted the importance of other mechanisms (Camos & Barrouillet, 2014; Barrouillet & Camos, 2015). In particular, the effect of articulatory rehearsal (cf. Baddeley and Hitch (1974)'s phonological loop) on WM span depends on the verbal/non-verbal nature of the stimuli used. To our knowledge, no formal description of how attentional refreshing and articulatory rehearsal work together to maintain information in WM currently exists in the TBRS literature.

We built our model within the cognitive architecture, ACT-R (Anderson, 2007). The comprehensive nature of the architecture allows for modeling an entire task from start to finish. Such a capability is imperative for our goal of a fully integrated model. The architecture is composed of various modules that specialize in different areas of processing, from more central functions like goal planning to more peripheral, perception and action functionality. The modules communicate with each other through limited capacity buffers that are each able to hold a single piece of information called a chunk. A production system, operating on if-then rules, coordinates the behavior of the modules. Only a single production may fire (i.e. be selected and executed) at a time. This limit, combined with the modules' buffers, comprises the bottleneck assumed by TBRS.

One module of ACT-R in particular, the declarative module, is especially useful for our purposes. It operates on retrieval requests where the production system sends the module a set of features as a cue. The module compares the request to the chunks in declarative memory and returns the chunk with the highest activation if it is above a threshold. One component of the equation used to calculate chunk activation, base-level learning (discussed later), implements the time-based decay and attentional refreshing assumptions of TBRS. In brief, the more time that has passed since a chunk was last retrieved, the less its activation will be, and the more times a chunk has been retrieved, the greater its activation will be.

A Benchmark Task

We chose an existing experiment from the literature, the third experiment from Barrouillet et al. (2007), as a benchmark for the model to meet. Using pre-existing data allows us to develop the model around less contentious criteria than data collected expressly for the purpose of fitting the model. It also affords us with the opportunity to determine the model's novel predictions *before* conducting new experiments.

Subjects completed a variant of the common complex span task where they were given a consonant letter to memorize followed by a series of digits (*distractors*), presented one at a time. Afterwards, they were to speak the list of to-be-remembered letters (*targets*). Subjects completed three trials for each list length, starting with one letter, and progressed to longer lists so long as they successfully recalled at least one of the three trials correctly. Every target was presented for 1500 ms, followed by a 500 ms delay, and then 6400 ms of distractors. CL was manipulated by changing the number of distractors (4, 6, and 8) presented during this fixed interval, effectively changing the pacing of the task. While the same stimuli were used in all conditions, CL was further manipulated by how subjects were instructed to respond to the distractors. In the parity condition, subjects were asked to respond whether the distractor was odd or even. In the spatial condition, subjects were asked to respond whether the distractor appeared in the top or bottom half of the display. Before the experiment began, subjects underwent training where they completed 96 distractor judgments (with feedback) and three trials each of one- and two-item lists in the manner and at the pace of their assigned experimental condition. WM span scores were calculated by dividing the number of correctly recalled trials for all lists by three (all-or-nothing unit scoring; Conway et al., 2005). Each of the six conditions exerts a different demand on the subject, so we should expect to observe different mean WM spans. The prediction of TBRS is that while the raw spans may differ, they should be the same once each condition is equated in terms of CL, and indeed this was what Barrouillet et al. (2007) found.

The Model

The general behavior of the model[1] can be organized into essentially three levels of priority. The highest priority productions support perception. Whenever a stimulus appears on the (virtual) display, the model moves its visual attention to encode it. Likewise, if something is available in the audicon, the model shifts its attention to it. Once the model attends to a stimulus, it then encodes the stimulus by retrieving its associated semantic information (i.e. chunk) from declarative memory. The type of the chunk retrieved dictates which of the intermediate-priority routines are followed next.

If an asterisk is presented, signifying the beginning of a new list, the model creates a new list-context in its goal buffer and sets a new goal to encode the next letter presented with an additional feature marking it as the head of the list. If a letter is presented, the imaginal module creates a new episodic representation of the target that includes references to the current temporal-context and list-context. If the word "recall" is presented, the model begins recalling and verbally reciting the target list. We further describe the details of this process later once maintenance has been addressed.

Distractor processing resembles a binary decision tree. At each step, the model decides between responding and retrieving more information. For example, when making parity judgments, the model may first blindly guess or wait until it has fully encoded the digit. Next, it may respond based on the number alone or retrieve the parity of the number. Then it may respond based on the number's parity or retrieve the response rule that connects the semantic concepts with their

[1] For a more thorough motivation and explanation of each of the model's assumptions and components, see (Glavan, 2017).

associated keyboard response. Once a response rule has been retrieved, the model faithfully uses it to make a key press. Thus, the model only makes errors by responding with insufficient information. More sophisticated decision making mechanisms could be included in the future, but this simple strategy is sufficient for our purposes here. The model uses the same strategy in the spatial condition but requires fewer retrievals; in fact it only requires retrieving the associated response rule.

The model learns the appropriate response mappings through utility learning during training. If the model responds incorrectly, it may learn to retrieve more information and respond later; if the model does not respond in time, it will be penalized and be more likely to respond earlier because a guess has a better chance of being correct than a lapse, which is always incorrect. In this way, the model should exhibit a speed-accuracy trade-off (observed in Barrouillet et al., 2007). In order to bias the model toward initially using its knowledge of the task instructions over learning by purely random guessing, we give the information-seeking productions greater starting utility than the early-response productions. Additionally, the model may continue to learn after training through production compilation, which may eliminate retrievals altogether by combining compatible productions. For example, the model may learn to associate a number's parity with its appropriate response directly, bypassing the need to retrieve a response rule (and reducing CL).

The model engages in maintenance, either attentional refreshing or articulatory rehearsal, whenever it is not busy. Attentional refreshing is implemented as simple retrieval from declarative memory. It uses two productions that leverage the remotely linked structure of the list. The first production initiates maintenance by requesting any target chunk. The second production uses the just retrieved item as the cue for a subsequent retrieval and loops until interrupted by a higher priority routine. In a similar fashion, recall begins by first requesting the head of the list, and then a second production vocalizes the just retrieved target while using it as a cue to retrieve the next item.

The maintenance and recall productions are able to successfully iterate the list by using the just retrieved item as a cue for the next item because of two terms that influence the activation of a chunk. First, partial matching penalizes chunks based on episodic/temporal dissimilarity – chunks that were encoded at more disparate times receive a greater penalty. We quantify the penalty using the logarithm of the temporal difference: $-\eta \cdot \ln\left(1+\left|\varepsilon_i-\varepsilon_{\text{requested}}\right|\right)$. Without compensation, this makes the just retrieved chunk the most likely to be retrieved next (preventing iteration through the list) and means that the consecutive target is as likely to be retrieved next as the previous target. Hence, we include temporal inhibition (Lebiere & Best, 2009) to penalize chunks based on how recently they were retrieved. Note, this does not force unidirectional refreshing. If an item were skipped the transposition error may be followed by a fill-in error because the skipped item has not been inhibited.

The component of the activation equation that implements TBRS's assumptions regarding attentional refreshing is the base-level learning mechanism. It treats every time a chunk is cleared to declarative memory (e.g., following retrieval) as a separate memory trace that decays as a power law. A chunk's base-level activation is the log-sum of its decaying traces: $\ln\left(\sum_{j=1}^{n} t_{ij}^{-\delta}\right)$. Thus, the longer it has been since a chunk was retrieved, the less likely it is to be retrieved again (recency effect), and the more often it has been retrieved, the more likely it is to be retrieved again (frequency effect).

We do not assume an explicit maintenance strategy (i.e. some fixed pattern like always starting from the beginning of the list). Instead, the temporal dynamics of decay, refreshing, and inhibition determine which item will resume maintenance after interruption.

We propose that articulatory rehearsal supports maintenance by establishing a more stable signal for cueing retrieval than the temporal dynamics of attentional refreshing allow. The audicon (cf. phonological loop, Baddeley & Hitch, 1974; echoic memory, Cowan, 1984) is managed by the aural module and is able to hold sensory information that the model heard in the last 3 seconds in the crude order it was heard. We suggest that the cognitive system can leverage the alternative temporal and structural constraints of this sensory memory register without a complex, CL-inducing strategy by leaving the articulated target information in its raw sensory form (i.e. without converting it to an episodic representation) and using the oldest sensory chunk available to directly cue retrieval, bypassing the episodic similarity penalty.

Maintenance, with articulatory rehearsal engaged, begins in the same way as previously described for attentional refreshing: a simple retrieval for any target chunk is requested. After one target has been retrieved, the next step depends on whether there is a chunk available in the aural buffer. If not, then another retrieval using the just retrieved item is requested (i.e. attentional refreshing), and if the vocal module is free, the just retrieved item is also subvocalized, loading the articulatory loop. If there is a chunk available in the aural buffer, and the vocal module is free, then the aural chunk is subvocalized and a retrieval of its corresponding target is requested. If the declarative module is busy (e.g., with the distractor task) but there is a chunk in the aural buffer and the vocal module is free, then the aural chunk is subvocalized without involving the retrieval mechanism, perpetuating the articulatory loop.

Simulation Study

The model has six free parameters: reward, inhibition decay, episodic selectivity, base-level constant, and two retrieval latency scaling parameters. All other parameters were fixed at architectural defaults. We fixed the starting utility of the information-seeking productions (see distractor processing above) to the reward parameter to avoid adding an additional free parameter.

As noted earlier, the bulk of the support for WM span as a

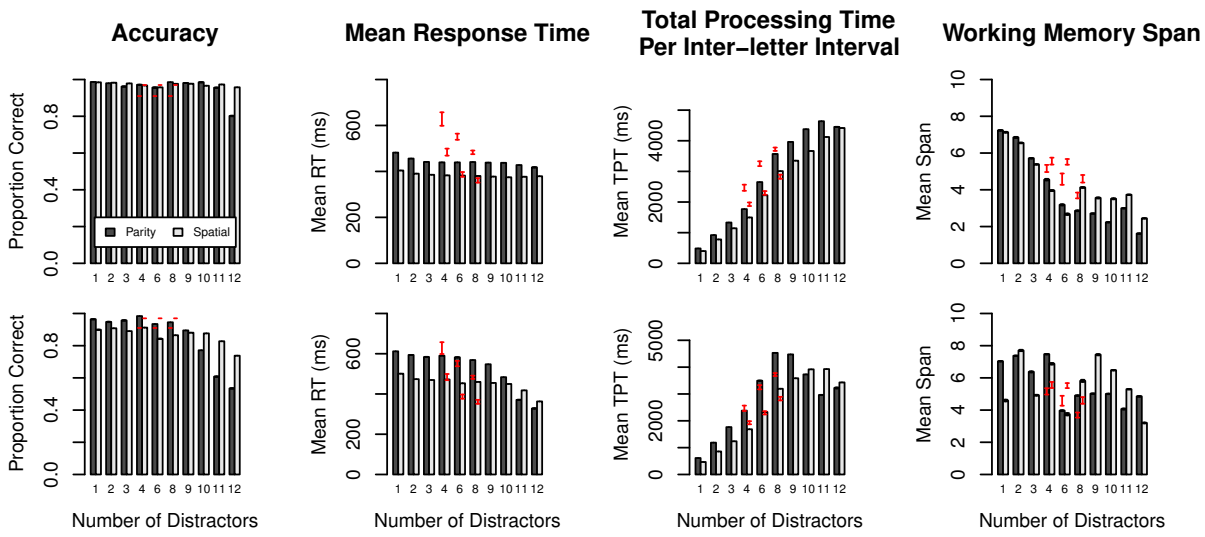


Figure 1: The three leftmost panels show accuracy, mean response time, and mean total processing time per inter-letter interval for the parity (darker bars) and spatial judgments (lighter bars). The far right panels show mean WM span. The top row comes from the attentional refreshing-only version of the model, and the bottom row comes from the version with articulatory rehearsal. Standard errors of the means from Barrouillet et al. (2007) are overlaid in red; note that Barrouillet et al. did not report accuracy variance or means by number of distractors so we plot only the mean difference by task.

linear function of CL comes from studies where spans were probed in the middle of the CL scale. To further test this hypothesis, we simulated the benchmark task (Barrouillet et al., 2007) in two different ways. In the first method, we forced specific amounts of CL on the model by preventing any refreshing or rehearsal productions from firing during the first portion of the inter-letter interval. We st allowed the version of the model with articulatory rehearsal to continue repeating items within the vocal-aural loop during this period as if the central bottleneck was obstructed by a non-verbal task. This provides a potentially undue advantage because the phonological code used to access items during rehearsal can remain intact in the articulatory loop.

In the second method, the model completed the parity and spatial judgment tasks to induce CL. Recall that Barrouillet et al. (2007) kept the inter-letter interval constant while using 4, 6, or 8 distractors to create three levels of CL; we add levels with 1, 2, 3, 9, 10, 11, and 12 distractors. In this version of the simulation, we estimate CL as the sum of distractor response times per inter-letter interval (Barrouillet et al., 2007). We expect there to be small differences between the two approaches because the predictor variables are not exactly the same, but the overall trend should persist. Lastly, because we are using lower levels of CL, some conditions should produce WM spans greater than those observed in the original study; therefore, we allowed the model to encounter target lists as long as 10 items, whereas the original study stopped at 7.

We used MindModeling@Home (Harris, Gluck, Mielke, & Moore, 2009) to speed up our extensive simulation of the large parameter space. We began fitting the model's free parameters by first enumerating a coarse grid of the six-dimensional parameter space with 10 repetitions of each condition. We used a weighted linear composite score from Glavan (2017) that uses multiple dependent variables (accuracy, mean response time, mean total processing time, mean span, and the slope of the regression of span on CL) to quantify the model's misfit to the human data (Barrouillet et al., 2007). In this step, we only considered the six experimental conditions common to the model and the human study. Based on these results, we fixed the reward, latency exponent, and latency factor parameters at 7.0, 0.3, and 0.7, respectively, because they reasonably approximated most of the accuracy and response time trends in the human data[2]. We then conducted a more focused, fine-grained enumeration of the three-dimensional parameter space using 20 repetitions. For this search, we used the RMSE of the predicted CL and span to the regression line from Barrouillet et al. (2011) as our measure of fit because in this paper we are more concerned with the model's ability to fit the proposed function of CL and because the remaining free parameters did not seem to affect accuracy or response time. It turned out that the results from the attentional refreshing and articulatory rehearsal versions of the model agreed on the best-fitting parameterization of inhibition decay (5.7), episodic selectivity (6.0), and

[2]For a more detailed exploration of the relationships among parameters, see Glavan (2017).

base-level constant (15), so we used this final parameterization to simulate 1000 runs of the model in the two CL and the two maintenance versions of the model. The accuracy, mean response time, mean total processing time per inter-letter interval (a proxy for CL because the inter-letter interval was constant), and mean WM span predicted by the attentional refreshing-only and articulatory rehearsal versions of the model are shown in Figure 1. Figure 2 shows the WM span as a function of CL.

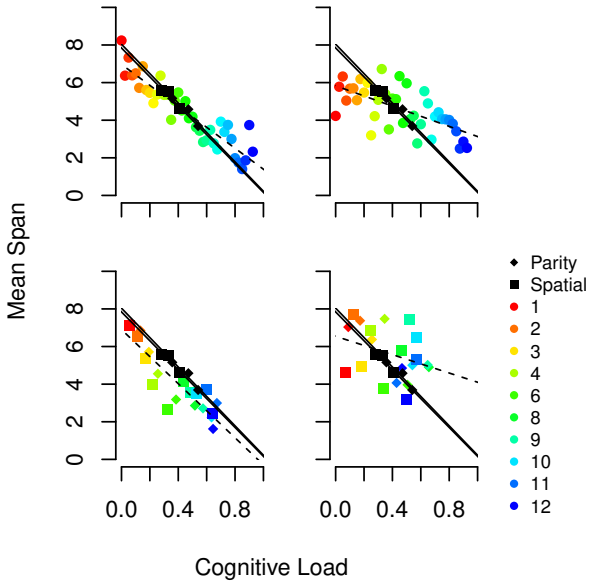


Figure 2: Mean WM span as a function of cognitive load. The attentional refreshing-only version of the model is presented on the left, and the version with articulatory rehearsal is presented on the right. The top two panels come from forcing specific levels of CL on the model, whereas CL in the bottom panels is estimated from performance on the parity (diamonds) and spatial (squares) judgment tasks. In the bottom panels, the color of the plotting symbols indicates the number of distractors processed between the presentation of each target. Results from Barrouillet et al. (2007) are plotted in black. Regression lines (solid for the human data, dashed for the model) are also included.

Discussion

As expected, the distractor results were similar regardless of whether articulatory rehearsal was included (Figure 1). Response times were approximately 100 ms longer for the articulatory rehearsal version because the model is repeating targets to itself while the declarative module is busy processing distractors. Thus, simply making articulatory rehearsal available as a cognitive strategy may increase CL.

The primary difference between these two models is in WM span. When CL is fixed (Figure 2, top panel), span is approximately linear for both models. The articulatory rehearsal version predicts a smaller effect of CL and resulted in more variation around the line. This shallower function may reflect the slower but more stable nature of the articulatory rehearsal signal for attentional refreshing. The maximum span at lower CL may be lower than the attentional refreshing version because the refreshing loop can cycle faster than the articulatory loop, but the span at higher CL may be higher than the attentional refreshing version because the direct retrieval of targets via the articulatory cue, which does not decay with increasing (non-verbal) CL is more able to refresh items than when the episodic similarity penalty is involved.

When CL was not fixed, and the distractor task performance was simulated (Figure 2, bottom panel) span results largely agreed with the forced-CL version. An interesting prediction of the model is that mean span may not always continue to decrease with increased processing pace as suggested by previous descriptions of TBRS. Consider the conditions of Figure 1 with the most distractors per inter-letter interval. Rather than predicting continuously increasing total processing time, the *integrated* model predicts that when the presentation rate of distractors is too high, the agent switches strategies in order to maximize performance (e.g., guessing, because fast guesses are better than guaranteed lapses). Critically, the model predicts that WM span can actually increase with regard to distractor pace in these conditions because the strategy change reduces the CL (Figure 2, lower right). While the majority of TBRS-related studies have assumed a homogenous processing strategy across conditions, our integrated model demonstrates that the relationship between processing and storage proposed by TBRS holds even when this assumption is violated.

Our results give support to the hypothesis that WM span is a linear function of CL with the following caveats. Figure 2 only reflects the model's ability to fit the regressions of Barrouillet et al. (2007, 2011). We cannot guarantee that other parameterizations of the model could not fit alternative span functions (e.g., S-shaped, etc.). A full analysis of the model's flexibility is needed (Roberts & Pashler, 2000). Second, one could reasonably argue that WM span in Figure 2 (top left) begins to asymptote. From the current efforts, it is difficult to determine if this is noise. If WM span does level off at maximum CL, we hypothesize that this minimum will depend on the overall task time. Under full CL, no maintenance takes place, so recall should be a function of only time-based decay.

A significant difference between the forced-CL and task-driven versions of the model is the temporal distribution of CL between to-be-remembered stimuli. In the forced-CL version, all of the CL occurs during the first portion of the inter-letter interval, whereas in the task-driven version, CL is dispersed across the entire interval. TBRS does not distinguish between these conditions because it treats CL as the overall proportion of processing time over total task time. In process models, such as ours, it becomes clear that the distribution of CL may interact with the observed WM span. For example, when CL is forced toward the first half of task time, span mea-

sures likely reflect the amount of information that could still be reinforced following the period of CL. When CL is forced toward the second half of task time, span likely reflects the surviving activation of items following their consolidation. De Schrijver and Barrouillet (2016) showed that free time (for maintenance) coalesced into a single interval has an advantage in terms of mean span over time that is distributed over distractors, which may help to explain the overall lower spans observed in the task-driven version compared to the forced-CL version. Our larger point, that CL is also a function of time, as opposed to a summary proportion like that proposed by Barrouillet et al. (2004), demands further exploration of the effect of the distribution of CL on WM span.

We tried to further explore the increased variability of the model compared to the human data, and while the model appears sensitive to the stimulus and subject variance, we have not been able to reproduce span functions with as little variance as the human data. One explanation may be the all-or-nothing scoring of span; Conway et al. (2005) has suggested that partial-unit scoring is less sensitive to individual variability. Indeed, the more recent literature seems to have begun to use the percentage of items correctly recalled over this measure of WM span (e.g., Camos & Barrouillet, 2014).

We hope to expand the model in the future to use ACT-R's spreading activation mechanism (i.e. association) and more sophisticated representations of context instead of partial matching. This may enable the model to produce intrusion errors and clustering effects (Farrell, 2012; see also Glavan, 2017).

In conlcusion, we provided a process-level, computational model that supports TBRS's prediction that WM span is a linear function of CL. We also propose a novel interpretation of articulatory rehearsal as a stable signal to guide attentional refreshing. We hope that this model will inspire future process-level consideration of WM. Real world cognition takes place on a dynamic, continuous timescale. We know that WM, its constraints, and concurrent cognitive load are each related to various forms of higher-level cognition (e.g., decision making, reasoning, and problem solving; Conway et al., 2005; Payne, Bettman, & Johnson, 1988). Our work lays the foundation for coordinating each of these disciplines within a single, integrated computational process model.

References

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

Baddeley, A. D., & Hitch, G. J. (1974). Working memory. *Psychology of Learning and Motivation*, *8*, 47–89.

Barrouillet, P., Bernardin, S., & Camos, V. (2004). Time constraints and resource sharing in adults' working memory spans. *Journal of Experimental Psychology: General*, *133*(1), 83.

Barrouillet, P., Bernardin, S., Portrat, S., Vergauwe, E., & Camos, V. (2007). Time and cognitive load in working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *33*(3), 570.

Barrouillet, P., & Camos, V. (2015). *Working memory: Loss and reconstruction*. Psychology Press.

Barrouillet, P., Portrat, S., & Camos, V. (2011). On the law relating processing to storage in working memory. *Psychological Review*, *118*(2), 175.

Camos, V., & Barrouillet, P. (2014). Attentional and non-attentional systems in the maintenance of verbal information in working memory: The executive and phonological loops. *Frontiers in Human Neuroscience*, *8*, 900.

Case, R., Kurland, D. M., & Goldberg, J. (1982). Operational efficiency and the growth of short-term memory span. *Journal of Experimental Child Psychology*, *33*(3), 386–404.

Conway, A. R., Kane, M. J., Bunting, M. F., Hambrick, D. Z., Wilhelm, O., & Engle, R. W. (2005). Working memory span tasks: A methodological review and users guide. *Psychonomic Bulletin & Review*, *12*(5), 769–786.

Cowan, N. (1984). On short and long auditory stores. *Psychological bulletin*, *96*(2), 341.

De Schrijver, S., & Barrouillet, P. (2016). *Consolidation and refreshing in working memory.* (Poster presented at the 57^{th} Annual Meeting of the Psychonomics Society)

Farrell, S. (2012). Temporal clustering and sequencing in short-term memory and episodic memory. *Psychological Review*, *119*(2), 223.

Glavan, J. J. (2017). *Exploring the time-based resource-sharing model of working memory through computational modeling* (Unpublished master's thesis). Wright State University. (`http://rave.ohiolink.edu/etdc/view?acc_num=wright149609967802364`)

Harris, J., Gluck, K. A., Mielke, T., & Moore, L. R. (2009). Mindmodeling home... and anywhere else you have idle processors. In A. Howes, D. Peebles, & R. Cooper (Eds.), *Proceedings of the ninth international conference on cognitive modeling.* Manchester, United Kingdom: University of Manchester.

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).

Oberauer, K., & Lewandowsky, S. (2011). Modeling working memory: A computational implementation of the time-based resource-sharing theory. *Psychonomic Bulletin & Review*, *18*(1), 10–45.

Payne, J. W., Bettman, J. R., & Johnson, E. J. (1988). Adaptive strategy selection in decision making. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *14*(3), 534.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? a comment on theory testing. *Psychological Review*, *107*(2), 358.

Balancing Confidence and Information Costs in a Diagnostic Reasoning Task

Tim Halverson (thalverson@gmail.com)
Oregon Research in Cognitive Applications, LLC.
Springfield, OR 97477 USA

Christopher Stevens (christopher.stevens.28@us.af.mil) **Chris Fisher (christopher.fisher.27.ctr@us.af.mil)**
Cognitive Science, Models & Agents, Air Force Research Laboratory
Wright-Patterson AFB, OH 45433 USA

Ashley Haubert (Ashley.Chafin@udri.udayton.edu)
University of Dayton Research Institute
Dayton, OH 45469 USA

Christopher Myers (christopher.myers.29@us.af.mil)
Cognitive Science, Models & Agents, Air Force Research Laboratory
Wright-Patterson AFB, OH 45433 USA

Abstract

Decision making requires balancing response accuracy and information acquisition costs. Computational cognitive models can be used to assess these trade-offs. Observed data and models of a multi-cue decision task are presented where information acquisition cost and environmental validity were varied. Willingness to respond, an aspect of confidence measured here with the number of cues viewed before responding, decreased when there was no cost and increased faster over time when the environment was consistent. Three variations of a model, based on Instance-based Learning Theory, are presented. The final model explains the majority of observed trends in response time, cues viewed, response accuracy, and reward. Decreased willingness to respond was explained with an increase in response threshold, and information acquisition cost as a decrease. Contrary to expectations, the consistent / no-cost condition required the highest threshold to fit the observed data. The implications and future directions are discussed.

Keywords: decision making; multiple cues; cue cost; cue consistency; confidence; ACT-R; Instance-Based Learning Theory

Introduction

A fundamental dilemma in decision making is balancing the trade-off between accuracy and cost. Searching for information can improve the likelihood of selecting the best alternative, but doing so can be costly in terms of time, resources, and opportunity (Chittka et al., 2009). If sufficiently confident, one should commit to a decision. If not, one should seek more information. Thus, it is important for a decision maker to be able to assess their confidence in a decision based on the current evidence. But how confident is confident enough? It is not currently well understood how people evaluate their confidence in a decision and how they use this confidence to guide their decision making, especially in complex environments where information search is possible.

Confidence is a broad term with many associated meanings in past literature. Our interest here is in confidence in a more limited sense: the strength of a person's belief in the value of a decision alternative. Thus, as confidence increases, a person should be less likely to check additional cues because they already believe they have found the best alternative. Two important factors that should moderate this relationship are environmental consistency and information search costs. More consistent environments should result in greater confidence because cues are more predictive of outcomes and therefore fewer of them are needed to make judgments with the same degree of accuracy. Information search costs should make a person more reluctant to seek additional information and therefore require lower confidence to commit to a decision. This relationship is analogous to threshold adjustment in drift diffusion models (Ratcliff et al., 2016).

In this paper, we present three models that explore the dynamics of decision confidence in a multi-cue diagnosis task that requires participants to decide which information to seek about a patient's condition. Each model is capable of self-terminating information search, determining the preference of cues, and choosing a response from two options. The models differed in terms of the how the decision threshold is determined; the threshold governs the amount of information gathered before committing to a decision. In the following sections, related literature is reviewed, the task environment is described, and results from an empirical study are described. Finally, the three models are described in detail and their abilities to explain the empirical data is discussed.

Related Literature

In this section, we briefly review literature relevant to the research questions highlighted above. First, we review a utility-based learning approach used in computational cognitive models called Instance-Based Learning Theory. Next, literature associated with modeling confidence within decision-making is reviewed.

Instance-Based Learning Theory

Gonzalez et al. (2003) proposed Instance-based Learning Theory (IBLT) to explain how decisions are reached in dynamic environments. In IBLT, decisions are based on an

accumulation of *instances* of interactions with an environment. These instances consist of the situation (relevant task features), the decision given the situation, and the reward received given the response. Decisions are made by determining which response has the greatest utility, which is based on the combined (i.e., blended) reward values from all instances that match the current decision context (Lebiere, 1999). This use of "utility" is different than production utility in ACT-R's production system. IBLT also proposes a decision process that includes (1) recognizing instances in the environment, (2) judging instances as either typical, which uses greatest utility mentioned above, or atypical, which uses heuristics to determine the utility, (3) deciding to search for more information or make a decision, and (4) gather feedback about the decision, whether that is provided or inferred.

IBLT has been applied to a variety of tasks, most notably repeated binary-choice tasks (Lejarraga et al., 2010). Models using IBLT do well at predicting a wide variety of effects of environmental consistency and payoff. In such tasks, the model makes a binary decision based on repeated experience from a single cue. To our knowledge, IBLT has not been applied to tasks in which multiple cues can be selected for each individual decision. Nor are we aware of any IBLT model that determines its own experience-based threshold for self-terminating information search. Models equipped with these capacities are reported here, in which the environmental consistency and the cost of revealing cues were demonstrated to affect participants' *willingness to respond*, measured by the number of cues viewed by participants before responding.

Confidence

IBLT operates on the simple and rational principle that one should always choose the response alternative with the best expected utility. However, there is uncertainty in many real-world environments, such that it is not obvious whether an agent has enough information to make an accurate comparison between alternatives. Here we apply IBLT to better understand how people make these information-search decisions.

Previous research suggests that people approach multi-cue decision-making by bringing to bear a set of ecologically valid heuristics designed to reduce the cognitive demand of a decision (Gigerenzer simple heuristics). According to one such heuristic, called Take-the-best, one should base one's decision on the cue with the highest validity (the conditional probability of an alternative given the value of the cue). If that cue doesn't point to one of the alternatives, then consider the next most valid cue, etc. According to this approach, a person's confidence in a decision is a function of the validity of the last cue checked. As more cues are checked, the validity of the last checked cue decreases, resulting in lower confidence (Gigerenzer, 1991). However, it is not clear how such heuristics might be learned or modified in new environments

Past work has shown that subjective confidence can be modeled as a drift diffusion process. Diffusion models represent evidence accumulation as a random-walk between one or more decision boundaries (Ratcliff et al., 2016). Each decision boundary represents one decision alternative. When the model reaches a boundary, it chooses the corresponding alternative. Pleskac & Busemeyer (2010) showed that allowing the accumulation process to continue after making a decision and re-sampling from it later produces a good account of subjective confidence, accuracy, and speed in two-alternative forced choice tasks. Diffusion models have been successful in modeling decision-making in a variety of domains (Ratcliff et al., 2016), but it is not clear how to scale such models up to complex environments requiring sequential information search and decisions.

Three-cue, Binary-response Decision Task

We used a multi-cue diagnostic choice task with a medical cover story (Myers et al., 2015). Participants assumed the role of emergency room doctors whose task is to assign patients to an intensive care unit (ICU) for the treatment of heart attacks or to a nurse's bed for observation. In order to make this determination, participants could check for the presence of three symptoms (cues): chest pains, irregular heart rhythms, and other symptoms. Each of the three cues had an associated probability of returning a True value in a given trial. The cue probabilities for a positive symptom were as follows: cue1 (Chest Pains) = 0.25, cue2 (Irregular Heart Rhythms) = 0.40, and cue3 (Other Symptoms) = 0.75. These cues were related to the patient's true state according to the following probabilistic rule: if cue2 (Irregular Heart Rhythms) and cue3 (Other Symptoms) are True, then the patient is having a heart attack; the patient is not having a heart attack otherwise.

This rule was valid on 80% or 90% of the trials, depending on the between-subjects experimental condition. At the start of each trial, none of the cues were visible to the participant. The participant could choose to check any combination of the three cues in order to decide where to send the patient. Cues could cost the decision-maker in points or be free of cost (only taking time to click, view, and encode), depending on the between-subjects experimental condition.

Feedback was provided after every trial in the form of points. Correct responses received 100 points and incorrect responses lost 100 points. If there was a cost to check cues, then any points spent to do so was deducted from this amount. Points received (or lost) during a trial were added to a running total that was visible to the participant throughout the experiment. The more points participants received, the better their compensation at the end of the experiment.

Method

Eighty students, 45 female and 35 male, ranging in age from 18 to 45 years old (mean = 22.95) from the University of Dayton participated in the experiment. Participants were paid $10 plus a bonus award of up to $10 depending on the participant's final score.

To determine the effects of information costs and environment consistency, a 2 (environment consistency: 90%, 80%) x 2 (cue cost: 0, 30) between-subjects factorial design was

used. Participants were randomly assigned to each of the four conditions. Each participant completed the same set of trials, the order of which was randomly sequenced across participants. Participants completed 267 trials. This number ensured that enough instances of the correct symptom-to-response mapping was experienced across the trials. Data from trials were averaged into blocks, where each block contained 30 trials except for the last (9th) block, which contained 27 trials.

Observed Data

Analyses of Variance were conducted on four key dependent variables: response accuracy, reward received, response time, and number of cues viewed. Each analysis used a linear mixed-effects model; information acquisition (cue) cost and environment consistency were between-subject fixed effects, block was a within-subject fixed effect, and participant ID was a repeated-measure grouping factor. We begin with results for response accuracy.

Response Accuracy. There was a significant main effect of cue cost, $F(1,683) = 9.57$, $p < 0.01$, and a significant main effect of environmental consistency, $F(1,683) = 444.53$, $p < 0.01$. Response accuracy was significantly higher in high environmental-consistency ($M_{High} = 80.7\%$; $M_{Low} = 63.4\%$) and in the no cue-cost groups ($M_{0-Cost} = 73.5\%$; $M_{HighCost} = 70.7\%$; see Figure 1, top-left).

Reward Received. There was a significant two-way interaction between cue cost and block, $F(1,683) = 17.875$, $p < 0.01$, where average reward over blocks increased at a faster rate in the high environmental-consistency condition than in the low environmental-consistency condition. There was a significant main effect of cue cost, $F(1,683) = 425.76$, $p < 0.01$, and a significant main effect of environmental consistency, $F(1,683) = 281.04$, $p < 0.01$. Reward received was significantly greater in high environmental-consistency ($M_{High} = 42.83$; $M_{Low} = 12.50$) and in the no-cost groups ($M_{0-Cost} = 46.28$; $M_{HighCost} = 8.17$; see Figure 1, top-right).

Response time. There was a significant two-way interaction between cue cost and environmental consistency, $F(1,683) = 42.28$, $p < 0.01$, where response times for the high environmental-consistency group were less than the low environmental-consistency group ($M_{HighConsist} = 4.40s$; $M_{LowConsist} = 5.27s$) in the no-cost conditions, but the same in the high-cost conditions ($M_{HighConsist} = 3.70s$; $M_{LowConsist} = 3.63s$). There was also a two-way interaction between cue cost and block, $F(1,683) = 17.87$, $p < 0.01$, where the high environmental-consistency groups decreased response times at a significantly faster rate across blocks than the low environmental-consistency groups. There was a significant main effect of cue cost, $F(1,683) = 78.94$, $p < 0.01$, a significant main effect of environmental consistency $F(1,683) = 9.177$, $p =< 0.01$, and a significant main effect of block, $F(1,683) = 17.07$, $p < 0.01$ (see Figure 1, bottom-left).

Cues Viewed. There was a three-way interaction between cue cost, environmental consistency, and block, $F(1,683) = 8.881$, $p < 0.01$. There were significant two-way interactions between cue cost and block $F(1,683) = 7.149$, $p < 0.01$), and between cue cost and environmental consistency, $F(1,683) = 57.322$, $p < 0.01$. There was a main effect of cost, $F(1,683) = 1126.289$, $p < 0.01$) and environmental consistency, $F(1,683) = 5.13$, $p = 0.02$; see Figure 1, bottom-right).

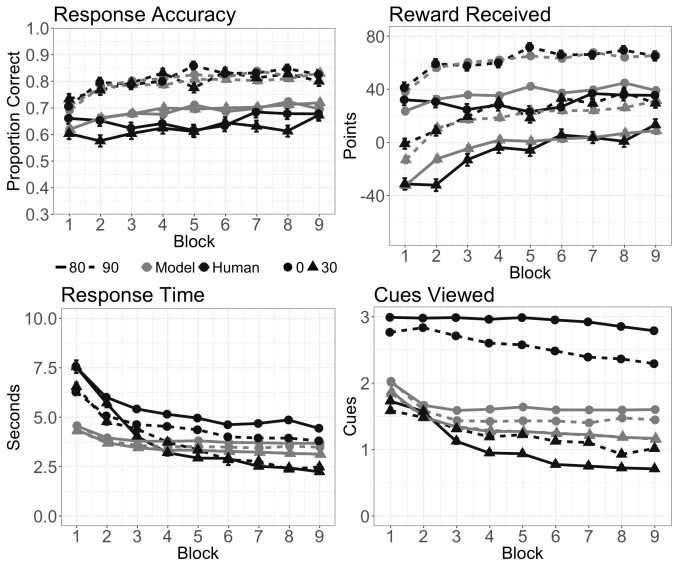


Figure 1: Observed (black) and base model (gray) response accuracy, response time, reward received, and cues viewed as a function of block, consistency (80%, 90%) and cue cost (0, 30). Error bars show ± 1 standard error.

Model Overview

A computational cognitive model was developed using the ACT-R 6 cognitive architecture (Anderson et al., 2004). This model incorporates many of the ideas from IBLT (Gonzalez et al., 2003) and built on a model of the same task with slightly different conditions (Myers et al., 2015).

The model interacts with an environment based on the task described above. The task environment presented the set of 267 stimuli-sets (three potential cues and an expected response) based on the cue consistency manipulation, shuffling the order of those stimuli-sets. The model completed each set of trials 50 times per condition, with the model and environment reset after each iteration.

As the model performed each trial, it built a representation (trial-instance chunks) within ACT-R's imaginal buffer. Each trial-instance chunk contained five slots on which decisions were based. Three *situation* slots represented values of selected cues. If the model checked a cue, the value was either "YES" (present) or "NO" (not present). If the cue was not checked, it was "UNKNOWN." One slot represented the model's *decision* for that trial of either "0" (nurses bed) or "1" (ICU). The final slot contained the reward from the response.

The model followed a straightforward process (see Fig-

ure 2, which proceeded as follows:

1. *Determine response threshold*: The model performed two blended retrievals, one for each potential response. Each retrieval determined the average utility (i.e., value) for a response based on the reward from *all* trial-instances in which that response was encoded. The highest blended value was used as the decision threshold (see step 5). In other words, future responses are expected to be better than the average of all previous correct and incorrect responses.

2. *Determine cue order*: Cue selection order was determined by three calls to the blending module, one for each cue. Each blend included all trial-instances in which that cue was encoded. The order in which cues were visited was determined by the descending order of the blended values.

3. *Select cue*: The model attended, clicked, and encoded the next cue, as determined by step 2.

4. *Determine the utility of each response*: A blend determined the utility of both potential responses based on all cues encoded thus far.

5. *Is a response's utility above threshold?* If one of the response utility determined in step 4 was above the threshold value determined in step 1 *minus* the cost associated with the number of cues revealed, then the model responded (i.e., go to step 6). If not, then the model selected another cue (i.e., go to step 3).

6. *Respond with the response with highest utility*: The model responded with the response of the blended trial-instance with the highest utility selected in step 6. After the response had been encoded, the model proceeded to the next trial and continued with step 1. The trial-instance from this trial was merged into declarative memory.

All blend retrievals used the arithmetic mean of feedback values and the default blending module parameters.

While the model learned all combinations of cues and responses through interaction with the environment, the model included a cue-counting heuristic that fired only if all three cues had been encoded *and* a response was not selected based on the utility of a decision. The model responded with "nurses bed" if one symptom were present, with "ICU" if three symptoms were present, and randomly if two symptoms were present. An analysis of the observed data supported this bias. When participants selected (and presumably attended) all three cues, they responded in a fairly predictable manner. When zero or one cue was true, 90% of all responses were "nurses bed." When all three cues were true, 95% of the all responses were "ICU." When two cues were present, responses were evenly split between the two.

All model parameters were fixed across the four experimental conditions. Three declarative memory parameters were varied to provide the best fit to the data. Base level learning, which controls the rate of declarative memory activation decay, was set to 0.4 (default 0.5). Base level constant, which is added to the activation of all declarative memory chunks, was set to 1.4. Retrieval threshold, which sets the minimum activation for declarative (or blended) retrievals, was set to 1.2 (default 0). Spreading activation was enabled for the imaginal buffer, which held the trial-instance chunk.

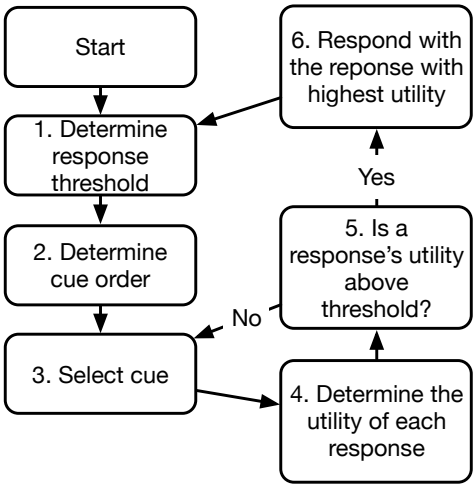


Figure 2: A representation of the model's main stages.

Figure 1 and Table 1 show the results of the model's predictions. This base model does a marginal job of predicting reward ($r^2 = .60; RMSD = 7.32$) and accuracy ($r^2 = .45; RMSD = 0.04$). However, the number of cues viewed ($r^2 = .52; RMSD = 0.71$) and response time ($r^2 = .93; RMSD = 1.22$) are not as differentiated by cost as is seen in the observed data.

The remainder of this section discusses two additional models that are variations of this base model. Each subsection identifies shortcomings found in the previous version of the model, proposes solutions, and shows the results.

Adaptive Threshold Model

The first variation recognizes that feedback and cost values used in the experiment are arbitrary values and may not be equally represented in people's cognitive processes across all conditions. The observed data shows that interaction between cue cost and consistency affected the number of cues viewed by the model. In the previous model, past response utilities and cue cost directly affected the number of cues viewed by the model by changing the response threshold. Perhaps these factors were weighted differently by the participants under different conditions.

A scaling factor was applied to the sum of the decision threshold and cue cost. This scaling factor was varied, by condition, between 0.2 and 3.0 in 0.2 increments to find a good fit. The threshold-scaling parameters were: (a) 1.0 for both high cue-cost conditions, (b) 1.2 for the high consistency / low cost condition, and (c) 1.5 for the low consistency / low cost condition. Changes to the declarative parameters established in the base model were varied slightly (BLL ± 0.1;

Table 1: R-squared and RMSD for the base, adaptive threshold, and fixed threshold models. Values shown are the mean of the individual condition fits.

	Model											
Metric	Base				Adaptive Threshold				Fixed Threshold			
	Cues	Acc	RT	Reward	Cues	Acc	RT	Reward	Cues	Acc	RT	Reward
r^2	.52	.45	.93	.60	.48	.44	.70	.59	.62	.44	.91	.61
RMSD	0.71	0.04	1.22	7.32	0.42	0.04	0.92	9.08	0.42	0.04	0.90	8.53

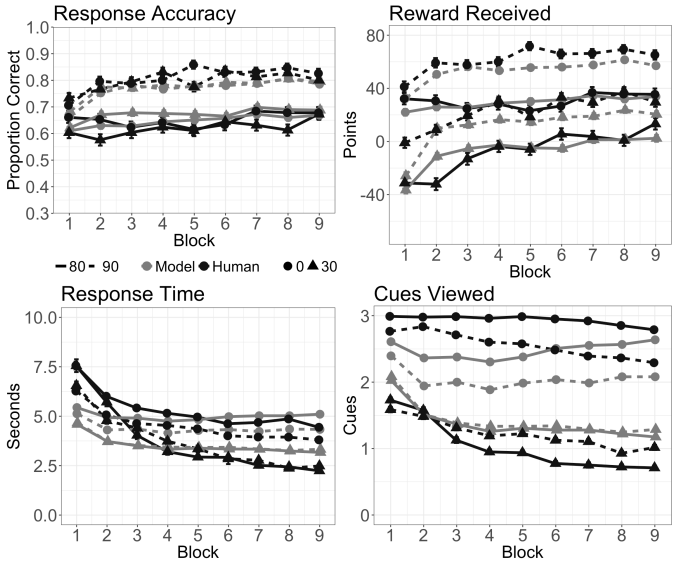


Figure 3: Observed (black) and adaptive threshold model (gray) response accuracy, response time, reward received, and cues viewed as a function of block, consistency (80%, 90%) and cue cost (0, 30). Error bars show ±1 standard error.

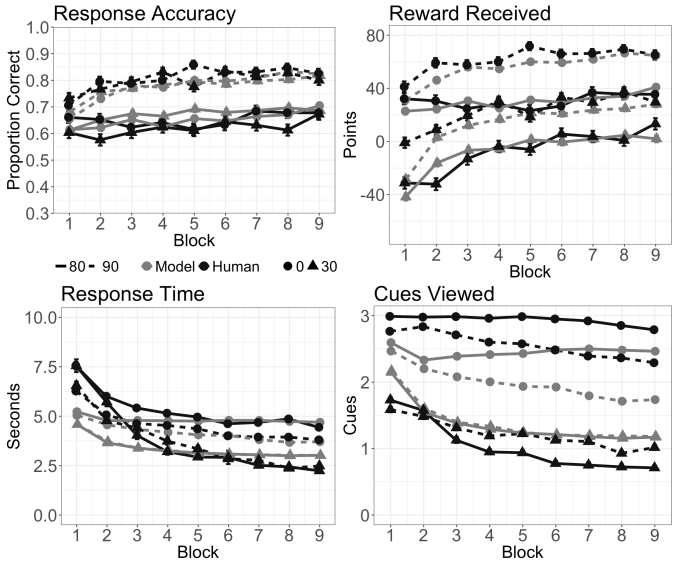


Figure 4: Observed (black) and fixed threshold model (gray) response accuracy, response time, reward received, and cues viewed as a function of block, consistency (80%, 90%) and cue cost (0, 30). Error bars show ±1 standard error.

BLC & RT ±0.6) and used only if the the change resulted in a substantial difference. The declarative parameters that differed from the base model were BLL (0.5) and BLC (1.8).

Figure 3 and Table 1 show the results of the model's predictions. The cues viewed were much more differentiated in the adaptive threshold model than in the base model ($RMSD = 0.42$ vs. $RMSD = 0.71$). However, the shape of those predictions were slightly worse ($RMSD = 0.48$ vs. $RMSD = 0.52$). In addition, the response time fit decreased substantially ($r^2 = .70$ vs. $r^2 = 0.93$). The next model includes an arguably simpler version of threshold modulation.

Fixed Threshold Model

The final variation simplifies the decision phase by using a fixed threshold for each condition, much like drift diffusion models of other decision tasks (Pleskac & Busemeyer, 2010). An adaptive threshold was used initially because it might account for learning trends in the observed data, as the threshold changed based on past performance. But perhaps people use fixed (or more stable) biases based on their level of confidence with the task. So next we explored the use of a static threshold that perhaps reflects a strategic shift based on the condition, rather than trial-by-trial variation.

A constant threshold was used for the decision phase. This value was varied, by condition, between 0 and 80 in increments of 5 to find a good fit. The threshold parameters were: (a) 20 for both high cost conditions, (b) 70 for the high consistency / low cost condition, and (c) 55 for the low consistency / low cost condition. All declarative memory parameters were identical to those used in the base model.

Figure 4 and Table 1 show the results of the model's predictions. Every fit metric improved or stayed the same, relative to the adaptive threshold model. Most importantly, the trends in cues viewed ($r^2 = .62$) and response time ($r^2 = .91$) improved substantially. The majority of the error remaining in the predictions lies in the intercepts for the number of cues viewed in the low-cost conditions. No additional straightforward changes were found that would make the model view three cues every trial, short of directly encoding that strategy. Regardless, the model does an excellent job of predicting the majority of the meaningful trends observed in the data.

Discussion

Accuracy in the observed and predicted data varies substantially by consistency. If the environment is less consistent, people are less sure of how to answer. When faced with a cost, as in the current task, people respond more quickly. Unexpectedly, people are only marginally (2.8%) more accurate when additional time is used to check cues in the no-cost conditions. In addition, response times are substantially slower (largely due to looking at more cues). The moderate accuracy improvement and substantial time decrements in the no-cost conditions means less reward, both in terms of points earned and financial compensation. So why are people acting somewhat irrationally when there are no cue costs?

The modeling suggests a straightforward explanation is that people hold information to a higher "standard" when there is no cue cost; the model requires higher and more consistent decision utilities when there is no cue cost. This higher information utility requirement does not vary at a fine-grained time scale. A static threshold predicts the results better than did a threshold that changes consistently based on recent experiences. Future research will need to explore other threshold dynamics, like thresholds that monotonically decrease across conditions. While such thresholds did not improve diffusion models of perceptual decision tasks (Voskuilen et al., 2016), they could prove useful here.

Further, when there are no cue costs *and* the environment is more consistent, a higher threshold is needed to explain the cues-viewed three-way interaction. This greater threshold bias (15 points or 27% higher) is counterintuitive strictly as a measure of confidence. If it were, then the bias would indicate a lower confidence in a more consistent environment. That is unlikely. A more likely explanation is that the threshold differences start to describe a mechanism that accounts for participants' strategic choice to withhold responses in uncertainty and when external pressures (i.e., cue cost) are applied. Perhaps the interplay between response utility (as a measure of familiarity) and the threshold bias (as a measure of willingness to respond) could produce a mechanism that explains decision confidence in this context. Further research is required to provide a more general model of confidence.

Conclusion

The cognitive modeling presented here suggests that IBLT provides a good framework to explain decisions when those decisions are made from experience and in situations when the cost of acquiring that experience varies. When there is cost to gathering information, that cost will lower the threshold used to select responses based on decision utilities. In addition, decision thresholds increase when there is no cost to gathering information. The precise function for determining that threshold increase is beyond the scope of this research, but we plan to address that shortcoming in the future.

The model developed here requires further validation with additional levels of cue cost. With only two levels of cost, the model may not properly characterize the variation in people's willingness to respond as a function of cost. In addition, the model should be validated with, and extended to make predictions of, subjective confidence. We are currently planning a study with the same task paradigm that will include additional cost levels and collect subjective confidence ratings.

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 supported by the Air Force Office of Scientific Research, grant 13RH06COR. The first author was supported by an appointment to the Postgraduate Research Participation Program at the U.S. Air Force Research Laboratory, 711th Human Performance Wing, Human Effectiveness Directorate, Warfighter Readiness Research Division, Cognitive Models and Agents Branch administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S. Department of Energy and USAFRL.

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.

Chittka, L., Skorupski, P., & Raine, N. E. (2009). Speed-accuracy tradeoffs in animal decision making. *Trends in Ecology and Evolution*, *24*(7), 400–407.

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

Lebiere, C. (1999). Blending: An ACT-R mechanism for aggregate retrievals. In *Act-r workshop.* George Mason University, USA.

Lejarraga, T., Dutt, V., & Gonzalez, C. (2010). Instance-based Learning: A General Model of Repeated Binary Choice. *Journal of Behavioral Decision Making*, *25*(2), 143–153.

Myers, C. W., Gluck, K. A., Harris, J., Veksler, V. D., Mielke, T., & Boyd, R. (2015). Evaluating Instance-based Learning in Multi-cue Diagnosis. In *Proceedings of the international conference on cognitive modeling* (pp. 198–199). Groningen, The Netherlands.

Pleskac, T. J., & Busemeyer, J. R. (2010). Two-stage dynamic signal detection: A theory of choice, decision time, and confidence. *Psychological Review*, *117*(3), 864–901.

Ratcliff, R., Smith, P. L., Brown, S. D., & McKoon, G. (2016, April). Diffusion Decision Model: Current Issues and History. *Trends in Cognitive Sciences*, *20*(4), 260–281.

Voskuilen, C., Ratcliff, R., & Smith, P. L. (2016). Comparing fixed and collapsing boundary versions of the diffusion model. *Journal of Mathematical Psychology*, *73*, 59–79.

Two Simple NeuroCognitive Associative Memory Models

Christian R. Huyck (c.huyck@mdx.ac.uk)
Department of Computer Science
Middlesex University
London UK

Yuhue Ji (YJ097@live.mdx.ac.uk)
Department of Computer Science
Middlesex University
London UK

Abstract

Human memory is associative and emerges from the behaviour of neurons. Two models, based on commonly used biological neural models are presented. The first model uses static synapses to approximate timing behaviour for a Stroop task with congruent conditions responding faster than incongruent conditions. The second model uses plastic synapses to learn a semantic net; it then duplicates the behaviour of a question answering task. This behaviour not only answers correctly, its times are similar to that of human subjects. These models are flawed in many ways, for instance, they use hundreds of neurons instead of the billions of neurons in the brain. They are thus not proposed as anything near a complete final model, but instead as early steps toward the development of more sophisticated neurocognitive associative memory models.

Keywords: Associative Memory; Cell Assembly; Hebbian Learning; Spiking Neuron.

Introduction

Human memory is associative in its nature (Anderson & Bower, 1973). Concepts are associated with related concepts, with, for instance, *Dog* associated with *Bone* and *Canine*. Similarly, human cognition is a product of the behaviour of the brain in general, and neurons in particular. Since associative memory is a key component of the human mind, and the mind is a product of the activity of neurons, developing simulations of associative memory in neurons is important.

One modern description of associative memory is the semantic net (Quillian, 1967). This symbolic representation has been widely used in knowledge representation schemes in Artificial Intelligence. Sub-symbolically, many connectionist and simulated neural systems have been developed to account for associative memory (for a review see (Lansner, 2009)). The authors are, however, unaware of any cognitive models of associative memory based on spiking neurons. As associative memory emerges from the behaviour of spiking neurons, such a model is important.

What is a good way to evaluate a cognitive model of associative memory? This paper describes two simple neurocognitive models of associative memory. The first accounts for the Stroop effect (Stroop, 1935) using static synapses. The second accounts for some subcategorisation hierarchy effects that subjects (Collins & Quillian, 1969) show in answering questions; this model makes use of plastic synapses, but has a rigid training regime.

These neuro-cognitive models are simple. They are based on simple neural models, and the second uses a simple Hebbian learning rule. As they are simple, they are flawed. These flaws are discussed, along with relatively simple mechanisms to add to build better neuro-cognitive models of associative memories. The conclusion includes future work in this area.

Literature Review

The task modelled in this paper is associative memory. There has been a great deal of associative memory modelling and psychological exploration of associative memory, and there is evidence that a crucial component of this memory at the neural level is the Cell Assembly (CA). Below this psychological and neuropsychological area is reviewed along with the neurobiological area of synaptic modification, and neural modelling.

Associative Memory

Human memory is associative. Concepts do not exist in isolation, but in a network of associations. A semantic net is a symbolic representation of this memory (Quillian, 1967). Collins and Quillian ran a psychological study that supports this (Collins & Quillian, 1969). In this study, subjects were given a statement such as *a canary is yellow*, and asked to say if it was true or false. It took subjects longer to respond false, but they also took longer if the fact was associated with a super-category of the item. They hypothesised that *bird* was the super-category of *canary* and *animal* the super-category of *bird*. It took longer to respond to *a canary has skin* than to *a canary can fly*, and the shortest was *a canary is yellow*. Similarly, the time to answer a question about direct hierarchical relations were longer the higher up the hierarchy. Shortest was *a canary is a canary*, followed by *a canary is a bird*, followed by *a canary is an animal*.

The Stroop effect (Stroop, 1935) is a consequence of associative interference. The original task has a colour name presented in a coloured ink. So, the word *blue* might be presented in *red* ink. One task is for the subjects to name the colour of the ink, in the example *red*. If the word and ink are congruent, the subjects respond faster and make fewer errors than if they are incongruent. This is a well known and a well studied phenomenon (MacLeod, 1991). Associative interference applies to many domains beyond colours and is a

window into human associative memory.

Cell Assemblies

The CA hypothesis is that the CA is the neural basis of, among other things, concepts (Hebb, 1949). A CA is a group of neurons that has relatively high synaptic connectivity, and relatively highly weighted synaptic connectivity. Thus, once the CA starts to fire, there is a cascade of firing that causes many of the other neurons in the CA to fire. This firing is the neural basis of a psychological short-term memory. The synaptic change required to make this connectivity is a long-term memory. Hebbian learning naturally leads to this type of structure.

There is a large community of researchers that, in essence, assumes that the CA hypothesis is correct, and the authors include themselves in this category. Though Hebb merely theorised the CA based on the limited biological evidence to hand, significant neurobiological evidence for CAs (see (Huyck & Passmore, 2013) for a review) has accumulated in subsequent decades. Indeed the authors are unaware of any evidence contradicting the CA hypothesis. However, the brain's complexity leaves the exact nature of CAs unclear.

Simulated spiking neurons are powerful computational devices. It is relatively simple to build systems based on spiking neurons that are incompatible with the CA hypothesis. These systems are suspect as models of human psychological behaviour.

Synaptic Plasticity

In computational neuro-biological circles, the most popular learning rule is currently spike timing dependent plasticity (Bi & Poo, 1998) (STDP). It is Hebbian; that is, the synaptic weight is increased if the pre-synaptic neuron tends to contribute to the post-synaptic neuron firing. There are many approaches to developing computational models of STDP.

There are also a wide range of learning mechanisms beyond STDP. One particularly useful system (Zenke, Agnes, & Gerstner, 2015) uses several learning rules to learn stable CAs. The network consists of both inhibitory and excitatory spiking neurons. In their simulations, STDP alone leads to unstable systems and stored CAs are erased over time. There is a rule that depresses synapses at high firing rates, and a related rule that increases synapses at low firing rates; both are based solely on one neuron, and are thus non-Hebbian. There are short term potentiation and depression rules. Metaplasticity rules are explored, and there are a range of time dynamics.

This enables the system to not only retain stored CAs, but to learn new CAs. This addresses the long standing neural stability plasticity dilemma (Carpenter & Grossberg, 1988). If learning remains on, so that new things can be learned, the old memories, stored synaptically, can be erased.

The range of synaptic weight modification mechanisms shows biology contrasts and complements mathematics. Simple mathematical rules help to explain the mechanisms, and can be implemented readily. They are usually approximations to biological mechanisms that are still poorly understood. In particular, the dynamic nature of the neural system, with spiking effecting synapse weights and synapse weights influencing spiking, makes it difficult to understand.

One simple rule is known as Oja's rule (Oja, 1982). This rule leads to the synaptic weight reflecting the likelihood that the post-synaptic neuron fires when the pre-synaptic neuron fires, their correlation. It has two components, an increase rule when the neurons co-fire, and a decrease rule when the pre-synaptic neuron fires, but the post-synaptic neuron does not. It can be modelled as the early part of equations 1 and 2.

$$\Delta_{+} w_{ij} = R * [(1 - w_{ij}) * 2^{(W_B - W_k)}] \quad (1)$$

$$\Delta_{-} w_{ij} = -R * [w_{ij} * 2^{(W_k - W_B)}] \quad (2)$$

In these equations R is the learning rate, and w_{ij} is the current synaptic weight. The exponential components of the equations are the compensatory modifiers, not used in Oja's rule and explained below. Using these rules, if the post-synaptic neurons fires 40% of the times when the pre-synaptic neuron fires, the weight will be approximately 0.4.

In addition to Oja's correlation component, the equations have a compensatory component that forces the total synaptic weight of a neuron toward a value W_B. The compensatory modifier is the exponential value at the end of the equations, and current synaptic weight of a neuron is W_k. The simulation below uses a pre-compensatory rule, where only the total synaptic weight of the pre-synaptic neuron is considered in the weight update.

The compensatory rule initially speeds learning, but also limits the synaptic weight. This limit prevents runaway synaptic growth. The authors have used these rules extensively (Huyck & Mitchell, 2014), and it has been suggested that compensatory processes are required for Hebbian learning (Zenke & Gerstner, 2017).

Simulating Neurons

There are many different computational models of neurons (see (Brette et al., 2007) for a review). One widely used class of model is a spiking point neuron, and one popular model is the adaptive exponential integrate-and-fire model (Brette & Gerstner, 2005).

The authors are involved in the Human Brain Project (HBP). To increase reproducibility and enable others to easily use models, the HBP uses a standard set of tools. Nest (Gewaltig & Diesmann, 2007) is commonly used to simulate neurons. PyNN (Davison et al., 2008) is used as middleware to specify the topology and eases the switch from one neural simulator or emulator to another.

Integrate and fire neurons are simple models of neurons, but they are widely used, and can accurately model firing behaviour of biological neurons. They are also computationally efficient to simulate.

One key question about cognitive behaviour is time. It has been noted that many connectionist schemes do not have time in them naturally (Elman, 1990). This is not the case with

simulations of biological neurons as they have a biological time course. Moreover, the time course of the neural behaviour is the same as the time course of the psychological behaviour it produces. So, while there are many connectionist models of associative memory (e.g. (Willshaw, Buneman, & Longuet-Higgins, 1969)), simulated biological neural models of associative memory are needed.

Stroop Model

There are many empirical findings that fall into the category of the Stroop effect (MacLeod, 1991). The cognitive explanation of the effect include horse-racing models based on different processing strength (Cohen, Dunbar, & McClelland, 1990), different perceptual acquisitions (Melara & Algom, 2003) and different selective attention (Roelofs, 2003). Different computational models have been proposed to simulate the Stroop effect based on different cognitive theories. The first connectionist model of the Stroop effect was built in a multi-layer perceptron and trained using supervised learning via back propagation (Cohen et al., 1990). In the same year, Phaf and colleagues developed a selective attentional model for the Stroop effect (Phaf, Van der Heijden, & Patrick, 1990). A more detailed model was built with sub-networks of sensory detection, motor response, attention control and habitual response (Kaplan, Şengör, Gürvit, & Güzeliş, 2007). In those models, the difference in response time was achieved by setting higher distraction on colour naming. A Hopfield network model for the Stroop effect was built and trained with combined patterns of attention and sensory inputs(Yusoff, Grüning, & Browne, 2011). The network converged to trained patterns based on the part-completion results of the attentional modulation in the testing set.

The authors have simulated the Stroop effect with simulated neurons using in Nest with PyNN.[1] Neurons representing ink colour, word, and outputs are modelled with leaky integrate-and-fire neurons. There are CA groups representing ink colour and word, and both are divided into two subgroups representing red or blue conditions. Excitatory connections within a CA spread activation leading to further activation. Inhibitory connections across conditions slow this spread. For instance, neurons in the CA for *blue ink* inhibit neurons in *red word* and vice versa. Synapses from the ink CAs excite their associated output neurons yielding the resulting time.

The response times, see table 1, were different across different conditions in colour-naming, which was interpreted as an interference of voluntary control (MacLeod, 2014).

Semantic Net Model

A plastic neural model of the question answering task (Collins & Quillian, 1969) was developed. The neural model was a variant of the adaptive exponential integrate-and-fire model (Brette & Gerstner, 2005). The learning mechanism was a compensatory Hebbian mechanism, see equations 1 and 2. The training regime was quite precise and the synapses were changed from plastic to static during the simulation, so that they no longer changed after training.

[1]The code for both models is available on http://www.cwa.mdx.ac.uk/NEAL/code/questionICCM.tar.gz.

Table 1: Response time in experiments and simulation

Conditions	Experiments (ms)	Simulation(ms)
Ink incongruent	795	660
Ink congruent	605	436

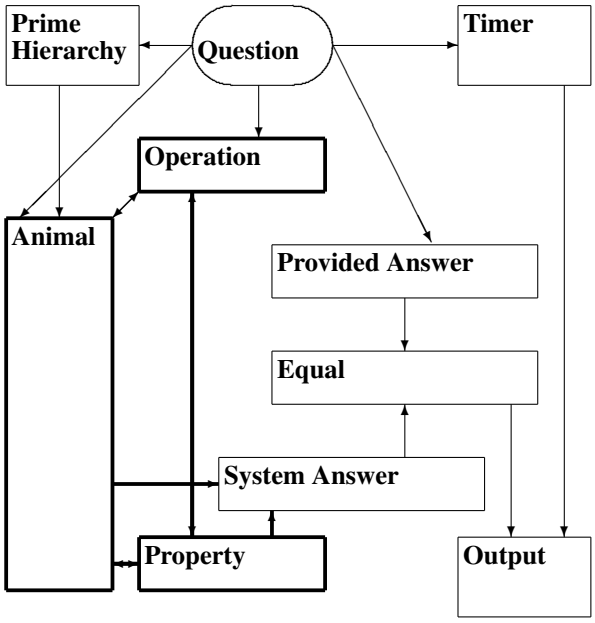


Figure 1: Gross Topology of the Question Answering Associative Memory. Boxes represent sets of neurons. Thick boxes and arrows are plastic. The oval represents the question with spike sources instead of neurons.

A pre-compensatory learning rule was developed in Nest as a synapse model. A pre-compensatory rule targets the total synaptic strength leaving a neuron, W_B in equations 1 and 2. Since the target and current strength need to be stored on the neuron, a new neural model was also developed in Nest. This modified the adaptive exponential integrate-and-fire model by including this constant and variable. The compensatory synapse changed the variable during synaptic weight change.

Figure 1 represents the final neural system. The system consists of three sets of neurons that have learned the semantic net. There is an animal inheritance hierarchy in the animal neurons. Associations are three way between animal, operation and property. For instance, if the *canary is yellow* association is stored, the synaptic weight between animal *canary* and operation *is* has increased weights, as do the synapses between *canary* and property *yellow*, and between *is* and *yellow*.

Training took place in two phases. The first phase learned the animal hierarchy and the second learned the associations.

Initially a well connected net of 200 neurons, the animal neurons, was trained to store 20 concepts and 19 direct hierarchical relations in a three level hierarchy (Animal (Amphibian) (Fish Shark Salmon Bass Pike) (Bird Canary Ostrich Robin Goose Pigeon) (Mammal Dog Cat Rat Bear Monkey Human)). A single concept was represented by 10 neurons with the first five being used for the hierarchy, and the second five used for associations. This was a simple model of a CA.

Neurons were stimulated externally in epochs. Each epoch went through a CA phase, a hierarchy phase and a one neuron firing phase lasting a total of 5000 ms of simulated time. The CA phase stimulated (to firing) each of the 10 neurons in a CA, and then inhibited the entire net after 40 ms. All 20 CAs were stimulated in turn, one after the other 50ms apart. This was followed by the hierarchy phase with each of the 19 direct hierarchical relations presented by stimulating (to firing) the first five neurons of each of the pairs; the entire net was inhibited after 40ms and the next pair was presented. This was followed by a period where each neuron was fired one at a time 11 ms apart. This allowed each synapse to apply the Hebbian decrease rule equation 2. Without this, the synapses between the neurons in the first half of a CA and the synapses between neurons in the second half only go up as they always fire together. This total epoch length was 20x50ms for the CAs, plus 19x50ms for the hierarchical relations, plus 200x11 for the one neuron firing phase, which was rounded to 5000. Hierarchical training took 30 epochs for 150000ms or 150 seconds. In the early epochs neurons fired once during CA and hierarchy presentation, but as the synaptic weight increased, they fired for several times. Inhibition 40ms after presentation allowed the next item to be presented.

The system was run in 1 ms time steps, and the compensatory mechanism considered the neurons co-firing if they fired in the same cycle or with the post-synaptic neuron firing within 10ms of the pre-synaptic neuron. In this manner, a simple CA for each animal concept is learned, and the hierarchical relations are learned.

The animal-animal synapses are saved, synapses with small values are pruned for efficiency, and they are loaded back in as static synapses for the second phase of training. In this phase, there were the 20 animal CAs in 200 neurons, 50 operation neurons, and 50 property neurons. The operation neurons were well connected internally, so that each neuron synapsed with every other neuron, and the property neurons were well connected internally. The second five neurons in each animal CA were well connected with both the operation and property neurons; so each had 100 additional synapses leaving them. Similarly each of the operation and property neurons had connections to those five animal neurons per CA, so each had 100 synapses to the animal neurons. The operation and property neurons connected to each other more sparsely with each connecting to 10 neurons (evenly distributed) in the opposite net.

These three nets were trained in two phases; the first learned the five CAs in the property and operation neurons. The second phase learned the five three way associations. After this the weights were saved.

These saved synaptic weights were loaded into the test system with the low weights pruned for efficiency. During testing, all synapses are static. The question is presented by external simulation. There are two types of question: animal relation property or animal isA animal. Both questions start the (neural) timer. The output neurons are two CAs, one for true and one for false. If the timer completes without the true CA coming on, the timer turns on the false CA, and that is the output. In this case, on means that the CA is firing persistently. The true and false output CAs have mutually inhibitory synapses.

For both types of question, the correct provided answer is stimulated; it fires persistently. Neurons in a CA can fire without causing the circuit to fire persistently; once a CA is firing persistently, it ignited. If the question is an animal relation property one, the animal and operation are stimulated. If there is a property associated, via learning, it becomes active. This then turns on the system answer. The equal net is a set of CAs that are only ignited if both the provided and system answers are firing. If there is an equal answer, it turns on the true output CA. The associations spread directly from the base level animal (canary), but can not spread until the super-level category (bird or animal) is activated. Thus it takes longer to retrieve these associations.

When the animal isA animal question, the second type, is asked, the animal, provided answer and timer are all stimulated. Operation is not stimulated, but the prime hierarchy CA is. This sends extra activation to all of the animals, which supports the spread of activation up the hierarchy. As before, the association is retrieved (or not) by the memory, in this case the animal, turning on the system answer CA.

Following Collins and Quillian (Collins & Quillian, 1969), the property associations are labelled P followed by the level, and the superset relation S followed by the level. The results are shown in table 2. Here S0 refers to the sentence *a pigeon is a pigeon*, and P3 refers to the sentence *a canary has skin*. The false sentences are labelled with False and are *a canary has gills* and *a pigeon is a fish*. The Collins column is the time reported in the paper (Collins & Quillian, 1969).

The associations are retrieved in the correct order by inheritance, but are clearly off time wise. It could easily be argued that the start times are due to input and output processes not accounted for by the model. So, 954ms, for instance, could be added to each of the systems times. Still, the timings are off significantly, with the system's times varying over less than 200ms and the subjects' over almost 500ms.

Discussion

The static Stroop model described above shows that it is reasonably easy to develop a spiking model of a particular associative memory. It is not entirely clear how well this mech-

Table 2: Associative Question Retrieval Time in ms.

Question Type	Collins Answer(ms)	System (ms)
P0	1300	51
P1	1380	61
P2	1460	62
P False	1450	235
S0	1000	46
S1	1170	59
S2	1240	130
S False	1400	235

anism scales. However, the number of neurons should scale linearly to the number of concepts. Similarly, if the associations are stored in synapses, and there are a constant number of associations for each concept, they should be storable in a topology like the sparse topologies of the brain.

While neural models have the advantage of parallelism, the real advantage to using neural systems is that they learn. The question answering model described above makes use of learning. While the Hebbian compensatory learning rule has a degree of biological plausibility, the presentation mechanism and shift from plastic to static neurons is clearly not plausible. One could argue that particular neuro-transmitters turn off plasticity, but the authors feel that is really stretching the metaphor. Instead, we view this model as a step toward more complete ones, and a very early step at that. It is using 10 neurons to represent a concept. The 10 neuron CA would persist indefinitely if not explicitly stopped. Once the neurons have stopped, they do not fire again unless stimulated from the environment, which is clearly biologically unrealistic. The words are stimulated directly from the environment; there is no attempt to read, and there is no attempt to actually ground these words in the environment. While it is clearly incomplete, the system does exhibit some symbolic properties of a semantic net. It also exhibits the right direction of timing for spread of activation of a reasonable cognitive model. It should be relatively simple to improve this so that it more accurately generates these times. This could be done by changing synaptic connectivity, or perhaps moving from a 5-5 CA in the hierarchical structure to a 10-5 CA. The neural parameters could be changed to support slower ignition.

However, improving the model by parameter fitting seems like an unpromising way forward for a significantly better neural associative memory. Instead the model could be improved by simple additions. For instance, the learned portion of the question answering model has no inhibitory neurons. Inhibitory neurons can also be used to reduce overall activation, but also support competition between concepts. The CA for fish is mutually incompatible with the CA for bird, so the two may have mutually inhibitory synapses.

The question answering model has also used synapses for associations. While synapses are clearly involved in associations, larger associated cell assemblies may share neurons. That is, neurons are involved in both CAs. When a CA ignites, neurons in it that are involved in other associated CAs are particularly efficient at priming those CAs and may even lead to their ignition. The authors have explore hierarchical CAs (Huyck, 2007) with shared neurons.

Another problem with the model, and indeed most neural models of learning, is that the neurons that learn are directly stimulated from the environment. Clearly, this is not the case in the brain with at most the sensory neurons being directly stimulated from the environment. Somehow learning must move from the sensory neurons into other areas. Again, the authors have made some progress on this (Huyck & Mitchell, 2014) using a Fatiguing Leaky Integrate and Fire neural model that spontaneously fires when it has not fired recently. Integrating spread into new areas into an associative memory, in addition to increasing biological plausibility, would also address input bandwidth problems of, for example, large neuromorphic machines.

The plastic question answering system only used one learning rule, though it did turn off learning. A better system might take advantage of several learning rules. In particular, the system could benefit from long and short term synaptic modification. The current rule changes the synaptic weight after the next firing, and that weight remains changed (though it of course may be modified again). This is not biologically plausible, as long-term synaptic modification is neither permanent (though it can last for months) nor instantaneous. The authors have explored short term dynamics, and hope to continue in the associative memory context. The long-term firing and synaptic dynamics need to be explored so the associative memory both stores old memories, and learns new memories.

Finally, the presentation mechanism in both models was to merely turn on stored neurons that represent symbols. This really is just a different way of using symbols. If the system could learn concepts from interaction with the environment, there would be scope for appreciably more complex concepts; this is the symbol grounding problem, and some progress on learning concepts can readily incorporate mechanisms for closely associating symbols with those concepts.

Conclusion

This paper has presented two neurocognitive models of associative memory. The first uses static synapses and duplicates the timing behaviour of performance on a Stroop task. The second uses a Hebbian compensatory synaptic modification rule to learn a semantic net. Performance on a question answering task is similar to behaviour of human subjects. Both models are implemented in leaky integrate and fire neurons.

These two models are simple, but it is hoped that they are just two early steps in the development of a more sophisticated neural associative memory mechanism. These models can be extended by the use of inhibitory neurons, supporting competition between CAs; associations including shared neurons, supporting a range of degrees of association; the use of neural models that support spread of CAs beyond neurons

that are directly activated by the environment, allowing the neural system to learn to use neurons that are not directly stimulated by the environment; and the combined use of multiple synaptic modification rules, providing improved flexibility with learning and more biological accuracy.

There will be two main strands in task development: symbolic bootstrapping and symbol grounding. Symbolic bootstrapping can use existing or newly developed symbolic semantic nets. These encodings can be learned by a neural system, and new associations can be learned by, for instance, interpreting text. Large semantic nets can be learned in large neuromorphic systems with millions of neurons, which can support exploration of CA and association dynamics.

Symbol grounding will be used for agents (virtual and robotic) that perform tasks. Initial bootstrapped semantic nets may provide memory, but new concepts and associations will be learned from the environment. This will address one of the key problems of AI.

The goal is to generate a substantially better neural associative memory. This memory will be evaluated on, among other things, the Stroop task, and the question answering task.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 720270 (the Human Brain Project).

References

Anderson, J., & Bower, G. (1973). *Human associative memory*. J. Wiley & Sons.

Bi, G., & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, *18:24*, 10464–10472.

Brette, R., & Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.*, *94*, 3637–3642.

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J., ... Destexhe, A. (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, *23*, 349–398.

Carpenter, G., & Grossberg, S. (1988). The art of adaptive pattern recognition by a self-organizing neural network. *IEEE Computer*, *21*, 77–88.

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

Collins, A., & Quillian, M. (1969). Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior*, *8(2)*, 240–247.

Davison, A., Brüderle, D., Eppler, J., Muller, E., Pecevski, D., Perrinet, L., & Yqer, P. (2008). PyNN: a common interface for neuronal network simulators. *Frontiers in neuroinformatics*, *2*.

Elman, J. (1990). Finding structure in time. *Cogntivie Science*, *14(2)*, 179–211.

Gewaltig, M., & Diesmann, M. (2007). NEST (neural simulation tool). *Scholarpedia*, *2*(4), 1430.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. J. Wiley & Sons.

Huyck, C. (2007). Creating hierarchical categories using cell assemblies. *Connection Science*, *19:1*, 1–24.

Huyck, C., & Mitchell, I. (2014). Post and pre-compensatory Hebbian learning for categorisation. *Computational Neurodynamics*, *8:4*, 299–311.

Huyck, C., & Passmore, P. (2013). A review of cell assemblies. *Biological Cybernetics*, *107:3*, 263–288.

Kaplan, G. B., Şengör, N., Gürvit, H., & Güzeliş, C. (2007). Modelling the stroop effect: A connectionist approach. *Neurocomputing*, *70*(7-9), 1414–1423.

Lansner, A. (2009). Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations. *Trends in neurosciences,*, *32(3)*, 178–186.

MacLeod, C. (1991). Half a century of the Stroop effect: An integrative review. *Psychological Bulletin*, *109:2*, 163–203.

MacLeod, C. (2014). The stroop effect. *Encyclopedia of Color Science and Technology*, 1–6.

Melara, R., & Algom, D. (2003). Driven by information: a tectonic theory of stroop effects. *Psychological review*, *110*(3), 422.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, *15*, 267–273.

Phaf, R., Van der Heijden, A., & Patrick, T. (1990). Slam: A connectionist model for attention in visual selection tasks. *Cognitive psychology*, *22*(3), 273–341.

Quillian, M. (1967). Word concepts: A theory of simulation of some basic semantic capabilities. *Behavioral Science*, *12*, 410-30.

Roelofs, A. (2003). Goal-referenced selection of verbal action: modeling attentional control in the stroop task. *Psychological review*, *110*(1), 88.

Stroop, J. (1935). Studies of inteference in serial verbal reactions. *J. of Experimental Psychology*, *18*, 643–662.

Willshaw, D., Buneman, O., & Longuet-Higgins, H. (1969). Non-holographic associative memory. *Nature*, *222*, 960–962.

Yusoff, N., Grüning, A., & Browne, A. (2011). Modelling the stroop effect: Dynamics in inhibition of automatic stimuli processing. In *Advances in cognitive neurodynamics (ii)* (pp. 641–645). Springer.

Zenke, F., Agnes, E., & Gerstner, W. (2015). Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks. *Nature communications*, *6*, 6922.

Zenke, F., & Gerstner, W. (2017). Hebbian plasticity requires compensatory processes on multiple timescales. *Phil. Trans. R. Soc. B*, *372(1715)*.

EEG classifiers can predict mind-wandering across different tasks

Christina Y. Jin (yi.jin@rug.nl)

Marieke K. van Vugt (m.k.van.vugt@rug.nl)

Jelmer P. Borst (j.p.borst@rug.nl)

Institute of Artificial Intelligence and Cognitive Engineering, Bernoulliborg, Nijenborgh 9
9747 AG Groningen, the Netherlands

Keywords: mind-wandering; single-trial ERP; support vector machine; sustained-attention-to-response task.

Background

Mind wandering is the process of thinking task-unrelated spontaneous thoughts while performing a certain task (Schooler et al., 2011; Smallwood & Schooler, 2015). The dynamics of mind wandering remain a puzzle, because it is difficult to track when someone is mind wandering based only on behavior. The goal of this study is to develop a machine-learning classifier that can determine someone's mind-wandering state online using electroencephalography (EEG) data.

Research showed that mind-wandering involves a decoupling from cortical processing of the task stimuli (Baird, Smallwood, Lutz, & Schooler, 2014; Kam & Handy, 2013; Kirschner, Kam, Handy, & Ward, 2012; Smallwood, Beach, Schooler, & Handy, 2008). We used several EEG markers related to this decoupling that had been suggested by previous research: P1, N1, P3 as the ERP measures, power in alpha (8.5~12Hz) and theta (4~8Hz) bands, as well as inter-site phase clustering (ISPC).

Method

We trained machine learning models on EEG features to classify the subject's current state in into mind-wandering or on-task. To examine whether mind-wandering signatures were task-specific, we also attempted to cross-classify between two tasks. The paradigms we used in this study were a sustained-attention-to-response task (SART) and a visual search task. While SART is common in mind-wandering research, we wanted to compare this to a task that relies more strongly on visual processing. In both tasks, probe questions asking for a self-report of the thoughts at that moment were randomly inserted and subjects' responses to the probes were used as classifier to label the preceding trials as mind-wandering or on-task. We analyzed six trials proceeding each probe, which account for roughly 30~36 seconds. This practice followed the assumption that the periodic fluctuations in attention might be supported by very low frequency (0.01~0.1Hz) coherence within the default mode network (Sonuga-Barke & Castellanos, 2007). Scalp EEG was recorded while subjects performed the tasks. The ERPs of interest were computed by a single-trial method, which is like a template-matching process (Bostanov, 2004; Bostanov & Kotchoubey, 2006). Furthermore, the clean EEG signal at A10, A19, B7 and C21 from a Biosemi 128 system (approximately PO7, Pz, PO8 and Fz in the 10-20 system) was band-pass filtered and Hilbert transformed to be decomposed into alpha and theta bands. Average power was computed for baseline (-400~0ms) and after stimulus onset (0~600ms) separately. Coherence was computed as the inter-site phase clustering (ISPC) between each combination of the selected electrodes in both bands.

We used support vector machine (SVM) as the training algorithm to use EEG markers to predict the current mind wandering state. Markers were z-transformed before entering the machine. The optimal regularization parameter C and γ were obtained through grid search. Considering individual differences in EEG patterns, model fitting was performed on each individual. If the data sample size was imbalanced between classes, we copied the cases from the minority class to make the training sample balanced (oversampling). Models were validated by leave-one-out cross validation (LOOCV). Performance was measured as prediction accuracy, sensitivity (true positive rate with positive defined as mind-wandering) and specificity (true negative rate with negative defined as on-task). Seventeen participants were included in the analysis.

Results

The reported mind-wandering rate during task performance varied across subjects (M=0.44, SE=0.04, range=0.15-0.82). The behavioral performance was disrupted when participants were in a mind-wandering compared to an on-task state. Response accuracy was significantly decreased in SART ($t_{(16)}$=-2.13, p=0.049, d=0.52) and marginally decreased in the visual search task ($t_{(16)}$=-1.87, p=0.079, d=0.46). Response time increased in the visual search task ($t_{(16)}$=2.20, p=0.043, d=0.53) while in SART there was no difference between conditions ($t_{(16)}$=-1.61, p=0.127, d=0.39).

Machine learning performance for each subject is shown in Figure 1. For the training and testing data based on the same task, the prediction accuracy ranged from 0.46 to 0.92 across individual models (M=0.62 for SART, M=0.65 for visual search task). For the across-task prediction, we

trained models from SART data and tested them on data of the visual search task (SART-VS) and vice versa (VS-SART). The obtained prediction accuracy ranged from 0.47 to 0.84 (M=0.61 for SART-VS, M=0.60 for VS-SART). The obtained sensitivity and specificity varied considerably across individuals. Sensitivity ranged from 0 to 0.97 with a mean of 0.49. Specificity ranged from 0.05 to 1 with a mean of 0.56. Overall, the prediction accuracy is above the chance level of 0.5. A t-test conducted between the obtained accuracy and 0.5 confirmed this difference ($t_{(67)}$=10.05, p<0.001).

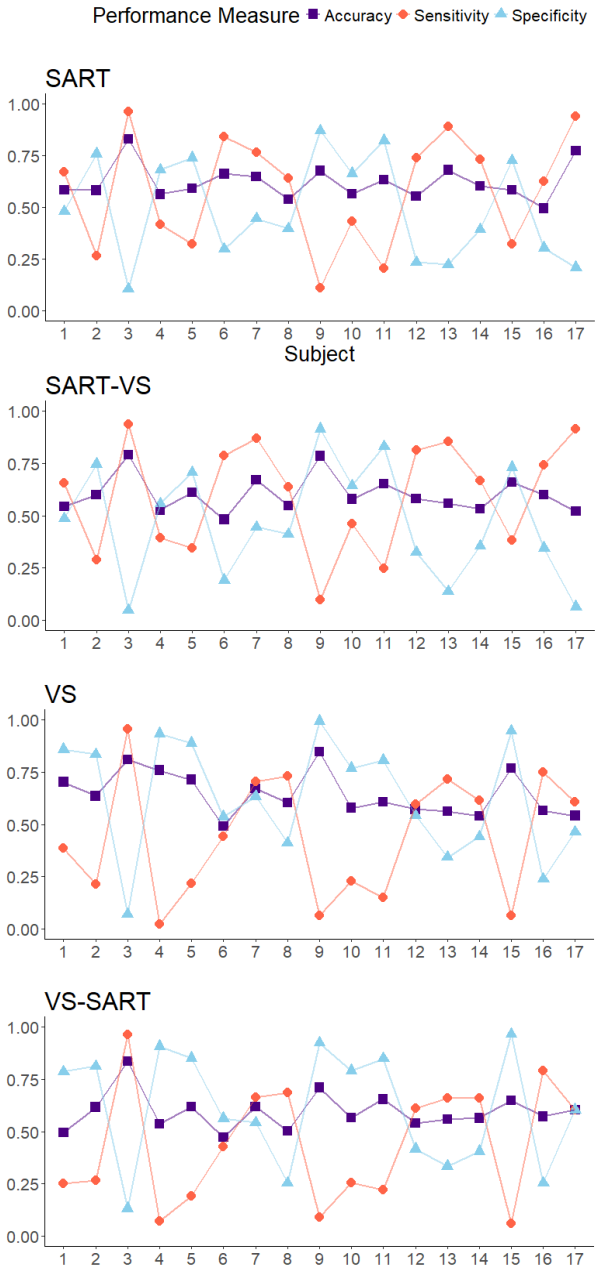


Figure 1: Machine learning performance of each individual data when training and testing them on the same task (SART, VS) or in an across-task way (SART-VS, VS-SART). SART = sustained-attention-to-response task, VS = visual search.

Discussion

Our study aimed to find task-independent neurophysiological EEG markers that can be used as the basis to differentiate mind-wandering from the on-task state. To achieve this goal, we had participants perform both an inhibition control task – SART – and a visual search task. On average, classification accuracy was above chance level. Moreover, even though classification accuracy was not very high, it was task general: it was possible to train on one task and use the obtained model to predict the data on another task. Results of this research confirm the potential to use EEG-based machine-learning classifiers to detect mind-wandering, without having to first train on the new tasks. In that way, the detection of mind-wandering could be less interfering and interruptive, allowing us to understand better when, how, and why mind-wandering occurs.

However, several cautionary remarks should be made. Although we were able to classify mind-wandering across tasks, general classification accuracy was relatively low. This is probably due to the difficult distinction we are trying to make; although there were small behavioral differences between the mind-wandering and on-task states, our participants did not stop performing their main task while mind-wandering, which made the two states highly similar. Unfortunately, this means that the current results cannot be directly used in clinical or industrial applications. If it were to be applied in some industrial application or medical practice like neurofeedback, performing some spatial filtering on EEG might be helpful because it extracts more discriminable EEG markers which might improve the prediction accuracy.

Second, the unbalanced sensitivity and specificity showed that the models were biased towards detecting one of the two classes. In a supplementary Spearman's rank correlation analysis between mind-wandering rate, sensitivity and specificity, we found that sensitivity was positively correlated to the mind-wandering rate during both tasks ($r_{(15)}$=0.89, p<0.001 in SART, $r_{(15)}$=0.83, p<0.001 in VS), while the specificity was negatively correlated to the mind-wandering rate ($r_{(15)}$=-0.77, p<0.001 in SART; $r_{(15)}$=-0.91, p<0.001 in VS). This indicated that the trained models were better at detecting the majority class. However, as we balanced the sample size in each class before training the SVM, this cannot be the result of learning the probability of each class. A possibility is that subjects held different standards when they decided their attentional states. Those who engaged more with the primary task might have a tendency to decide their momentary attentional state as on-task. On the contrary, those who engaged more with the mind-wandering process might tend to report off-task thinking. The blurred line between on-task and mind-wandering state when giving self-reports might cause the data to be imprecisely labeled, which further influenced the machine learning result.

To sum up, our research shows the potential to predict mind-wandering using interpretable neurophysiological EEG markers combined with machine learning. The classifier is task-independent because we achieved prediction accuracy over chance level in across-task predictions. Results of this study can be used to track mind-wandering more continuously in experimental settings.

References

Baird, B., Smallwood, J., Lutz, A., & Schooler, J. W. (2014). The decoupled mind: Mind-wandering disrupts cortical phase-locking to perceptual events. *Journal of Cognitive Neuroscience, 26*(11), 2596-2607. doi:10.1162/jocn_a_00656

Bostanov, V. (2004). BCI competition 2003-data sets Ib and IIb: feature extraction from event-related brain potentials with the continuous wavelet transform and the t-value scalogram. *IEEE Transactions on Biomedical engineering, 51*(6), 1057-1061.

Bostanov, V., & Kotchoubey, B. (2006). The t-CWT: a new ERP detection and quantification method based on the continuous wavelet transform and Student's t-statistics. *Clinical Neurophysiology, 117*(12), 2627-2644.

Kam, J. W. Y., & Handy, T. C. (2013). The neurocognitive consequences of the wandering mind: a mechanistic account of sensory-motor decoupling. *Frontiers in Psychology, 4*, 725. doi:10.3389/fpsyg.2013.00725

Kirschner, A., Kam, J. W. Y., Handy, T. C., & Ward, L. M. (2012). Differential synchronization in default and task-networks of the human brain. *Frontiers in Human Neuroscience, 6*, 139. doi:10.3389/fnhum.2012.00139

Schooler, J. W., Smallwood, J., Christoff, K., Handy, T. C., Reichle, E. D., & Sayette, M. A. (2011). Meta-awareness, perceptual decoupling and the wandering mind. *Trends in Cognitive Sciences, 15*(7), 319-326. doi:10.1016/j.tics.2011.05.006

Smallwood, J., Beach, E., Schooler, J. W., & Handy, T. C. (2008). Going AWOL in the brain: Mind wandering reduces cortical analysis of external events. *Journal of Cognitive Neuroscience, 20*(3), 458-469. doi:10.1162/jocn.2008.20.3.458

Smallwood, J., & Schooler, J. W. (2015). The science of mind wandering: empirically navigating the stream of consciousness. *Annual Review of Psychology, 66*, 487-518.

Sonuga-Barke, E. J. S., & Castellanos, F. X. (2007). Spontaneous attentional fluctuations in impaired states and pathological conditions: A neurobiological hypothesis. *Neuroscience & Biobehavioral Reviews, 31*(7), 977-986. doi:https://doi.org/10.1016/j.neubiorev.2007.02.005

Integrating Emotional and Rational Cognition

William G. Kennedy (wkennedy@GMU.Edu)
Center for Computational Social Science, George Mason University
Fairfax, VA 22030 USA

James C. Thompson (jthompsz@GMU.Edu)
Department of Psychology, George Mason University
Fairfax, VA 22030 USA

Abstract

It has long been understood that cognition involves at least a rational side and an emotional side. However, approaches to how these are integrated in our minds has not been done well. We propose an integration approach that has the emotional side affect the rational process and we discuss how this will be done building on the ACT-R architecture.

Keywords: Cognitive architectures; rational cognition; beyond rational cognition.

Introduction

At least since Plato's charioteer with two horses representing man's two natures have we been dealing with how both of these natures are managed within our minds (cite). We study rational cognition primarily because it is the classic, reliable, and (hopefully) repeatable cognition we see in experimental psychology laboratories and typically model in this conference. We build models in ACT-R (R for rational) (Anderson and Lebiere, 1998) and also as Bayesian models and neural network models to study rational cognition.

However, we know that emotions, emotional cognition, affect, and other forms of "beyond rational" reasoning also play a large role in determining behavior. Emotion is often discussed as a "cognitive moderator" (see Belavkin, Ritter, and Elliman, 1999). However, emotions seem to have a more important role in cognition that just as a "moderator." Emotions might be necessary to come to a decision as Damasio reports on two patients, Phineas Gage and "Elliot" (1994, pg 39). There he tells the story of the modern Phineas Gage patient, Elliot, with apparently no damage to his intellect or any "neurological dysfunction", but the inability to make a decision. The problem was that he could not decide between alternatives and would continue to consider options long after it was reasonable.

We do not have this effect in our cognitive architectures because we require them to make a decision. That seems to presume the emotional contribution to regulate the thinking so as to make a decision in a reasonable time frame. That last step, to break out of the exploration of possibilities and decide, seems to be a necessary emotional component of thought.

Current Approaches

The current approaches to integrating rational and emotional cognition vary from no integration, to some involvement of emotion in rational cognition, to selecting which form of cognition dominates. Each will be briefly reviewed.

Emotion in ACT-R

The R of ACT-R is rational. However, there have been models of stress and physiological conditions that provide clues to how emotion might be integrated into the ACT-R framework. The earliest papers on ACT-R and emotions (Belavkin, Ritter, and Elliman, 1999; Ritter, Belavkin, and Elliman, 1999; and Belavkin, 2001) focused on emotion as representable within the ACT-R architecture, as the gain in performance associated with productions. The last referenced Yerks-Dodson (1908) and the positive and negative effects of stress on performance. Cochran, Lee, and Chown (2006) discussed emotions as appraisal and arousal levels affecting memory activation. Stress and performance was modeled by Ritter, Schoelles, Klein, and Kase (2007) who used overlays to adjust ACT-R parameters. Dancy (2013) connected a model of human physiology to ACT-R and modified ACT-R parameters based on the physiology model. In last year's cognitive modeling conference (ICCM 2017) there were four papers addressing emotion and cognition, related to attention, attachment, rumination, and the relationships among emotion, mood, and personality. None of these treatments of emotion present a change in the ACT-R theory or architecture in that they all include handle emotion within the current architecture. We are proposing changing the architecture to address emotional aspects of cognition.

Emotion in Soar

Addressing emotion with Soar has been discussed as far back as feedback from an exercise of Soar agents in 1997 (Laird, et al. 1998). That feedback lead to a research effort to include emotion in models of agents used in teaching (Gratch and Marsella, 2004), but that was a separate part of the architecture, not integrated. John Laird himself notes that so far, emotion has been used in Soar only to establish the reinforcement reward and that incorporating emotion within an architecture could be important to future designs (Laird, 2012).

Emotion in Agent-Based Models

Agent models in computational social science have started to incorporate emotions. Epstein's "Agent_Zero" learns to respond to fear (Epstein, 2014) and the PECS architecture (Schmidt, 2002) selects either physiological, emotional, cognitive (rational), or social cognition options based on situational analysis. This is a selection, not an integration, but is considering the appropriate forms of cognition.

Architectural Approach

Our approach is architectural in that we are inspired by the brain's modular processing of emotions and we propose to change how the ACT-R theory and architecture functions, not just how a model operates. We propose three aspects of emotional impact on the normal (rational) processing of productions, the core of a cognitive architecture's rational processing.

Changing the Processing of Conditions

The left-hand side of a production or rule is the set of conditions under which the rule would fire. The basic design is the same for both ACT-R and Soar and in both, the ordering of the conditions, which all must be true for the rule to fire, does not matter. (The common model does not get to this level of detail, yet (Laird, Lebiere, and Rosenbloom, 2017). Our idea is that in an elevated emotional, or potentially any stressful state, we may not test all the conditions to get to the action. If the conditions were ordered in some way, then any production could be processed differently based on the level of stress by reducing the number of conditions evaluated and the rest ignored, i.e., handled as if they were true. The level of stress would determine how many or which of the conditions, would be processed: more stress, fewer conditions tested.

Interfering with Memory Retrievals

In ACT-R, a memory reference requires two productions to fire. The first makes a request declarative memory to recall a fact from memory and the second processes the retrieved item. Under high levels of stress, this two-step process could be disrupted by a change in the goal buffer such as the loss of some or all attributes and/or values during the latency in the recall process, similar to forgetting why we walked into a room… Another approach towards the same effect would be for the memory retrieval process to fail to make any retrieval. This would make memory-based productions unreliable during periods of high stress.

Interfering with Normally Sequential Rules

Finally, another architectural effect we propose is that multi-step rule-based strategies may not be successfully completed even though the necessary are productions are available in procedural memory and the external environment would support their firing. This could be accomplished in at least two ways. The first is that the system could forget where it is in the sequence and stop to reconsider. Second, it could skip steps (sequential rules) to get to the action faster. This also seems architectural in nature and could be implemented by unreliable attributes and values in the goal buffer changing between production firings or by implementing a series of rule compilations before they have matured through the normal ACT-R rule complication process for combining a sequence of steps. This could explain a model getting to an action involving the real world faster.

Integrated Theory Development

While these proposals might seem plausible, our goal is to promote discussion of the rightful place of emotion in cognitive architectures. We will discuss potential implementations and experiments at the conference.

References

Anderson, J. R., and Lebiere, C. J. (1998) *The atomic components of thought*. Psychology Press.

Belavkin, R. V. (2001). The Role of Emotion in Problem Solving. In Proceedings of the AISB'01 Symposium on Emotion, Cognition and Affective Computing (pp. 49–57). Heslington, York, England.

Belavkin, R. V., Ritter, F. E., & Elliman, D. G. (1999). Towards including simple emotions in a cognitive architecture in order to fit children's behaviour better. In Proceedings of the 1999 Conference of the Cognitive Science Society. Mahwah, NJ: Erlbaum.

Chong, R. (1999). Towards a model of fear in Soar. In *Proceedings of Soar Workshop* (Vol. 19, pp. 6-9).

Cochran, R. E., Lee, F. J., & Chown, E. (2006). Modeling emotion: Arousal's impact on memory In proceedings of the 28th Annual Conference of the Cognitive Science Society (pp. 1133-1138). Vancouver, British Columbia, Canada.

Damasio, A. R. (1994). Descartes' error: Emotion, Reason, and the Human Brain. Random House.

Dancy, C. L. (2013). "ACT-RΦ: A cognitive architecture with physiology and affect." Biologically Inspired Cognitive Architectures 6: 40–45.

Epstein, J. M. (2014). *Agent_Zero: Toward Neurocognitive Foundations for Generative Social Science: Toward Neurocognitive Foundations for Generative Social Science*. Princeton University Press.

ICCM 2017 (2017, July 23) ICCM2017: 15th International Conference on Cognitive Modeling. Retrieved from http://iccm-conference.org/2017/

Gratch, J., & Marsella, S. (2004). A domain-independent framework for modeling emotion. *Cognitive Systems Research*, *5*(4), 269-306.

Laird, J. E. (2012). *The Soar cognitive architecture*. MIT press.

Laird, J. E., Coulter, K. J., Jones, O. M., Kenny, P. G., Koss, F., & Nielsen, P. E. (1998). Integrating intelligent computer generated forces in distributed simulation: TacAir-Soar. In *in STOW-97." Proceedings of the 1998 Simulation Interoperability Workshop*.

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).

Ritter, F. E., Belavkin, R. V., & Elliman, D. G. (1999). Affective Computing: The role of emotion in human-computer interaction. In A. Monk, A. Sasse, & A.Crerar (Eds.) Proceedings of the British HCI Group one-day meeting in conjunction with University College London.

Ritter, F. E., Schoelles, M., Klein, L. C., & Kase, S. E. (2007). Modeling the range of performance on the serial subtraction task. In Proceedings of the 8th International Conference on Cognitive Modeling. Lewis, R. L., Polk, T. A., Laird, J. L., (eds.). 299-304. Oxford, UK: Taylor & Francis/Psychology Press.

Schmidt, B. (2002) Modeling of Human Behaviour: The PECS Reference Model. Proceedings 14th European Simulation Symposium. A. Verbraeck, W. Krug, eds.

Yerkes, R. M. & Dodson, J. D. (1908), 'The relation of strength of stimulus to rapidity of habit formation', *Journal of Comparative and Neurology and Psychology* 18, 459–482.

Visual Search without Selective Attention: A Cognitive Architecture Account

David E. Kieras (kieras@umich.edu)
Electrical Engineering & Computer Science Department, University of Michigan
2260 Hayward Street, Ann Arbor MI 48109-2121, USA

Abstract

A key phenomenon in visual search experiments is the linear relation of RT to the number of objects to be searched. The dominant theory of visual search claims that this is a result of covert selective attention operating sequentially to "bind" visual features into objects. However, a cognitive architectural model shows that this result can be easily obtained from basic visual mechanisms, eye movements, and a simple task strategy. No selective attention mechanism is needed.

Keywords: cognitive architecture, visual search; cognitive modeling; eye movements

Introduction

A myriad of visual search experiments using an especially simple visual search task have been published since the seminal work of Triesman & Gelade (1980) was advanced by Wolfe and his coworkers, starting with Wolfe, Cave, & Franzel (1989). In this task, subjects view a display containing several objects, and decide whether a specified *target* object is present or not, and make a corresponding keystroke response. The main independent variable is the number of objects on the display (*set size*), and the main dependent variable is the reaction time (*RT*), the time to make the response. Normally the target is present half the time (*positive* trials), and absent the other half (*negative* trials). Additional independent variables are the visual properties specified for the target, and the logical form of the specification.

The data typically produced by this task show a roughly linear increase in RT with set size, with negative trials producing a slope about twice as steep as positive trials. Different visual properties and target specifications produce positive trial slopes ranging from essentially zero (e.g. the target is a single red bar among green bars) to roughly 50 ms/item or more (e.g. a specific detailed shape among similar detailed shapes). The fact that the negative trial slope is about twice that of the positive trial slope suggests that a serial self-terminating search process is involved.

Covert Attention Theory of Visual Search

An obvious idea is that subjects move the eyes to each item sequentially to perform the search. However, the typical slopes observed are much faster than eye movements would allow. This discrepancy underlies the basic theoretical claim originally made, and still dominant in this literature, that the sequential search is done not by overtly moving the eyes, but instead by covertly moving selective attention from one object representation to another. This *covert selective attention* theory of visual search appears to have its roots in Neisser's (1967) assertion, based on extremely early computer vision concepts, that "focal attention" is necessary to bind together primitive features into a visual object; this attention-based "binding" operation was advanced in Triesman & Gelade (1980), and has been tremendously influential since then.

However, the dominance of the covert attention binding hypothesis has led to a remarkable dismissal of the role of retinal nonhomogeneity (i.e. the high resolution of the fovea compared to peripheral vision) and the eye movements that are required as a result (see Findlay & Gilchrist, 2003, for discussion). In fact there is little or no mention of such issues in Neisser, nor in the mainstream of visual search research descended from Treisman and Wolfe, even though many studies have demonstrated their relevance (e.g. Carrasco & Frieder, 1996; Wertheim, et al., 2006; Zelinsky & Sheinberg, 1995). It has even been claimed that the RT effects are the same regardless of whether or not eye movements are made. However, a full survey of the relevant literature, too lengthy to include here, shows that this claim is contradicted not only by empirical results, but also logical, methodological, and substantive problems. Simply put, eye movements should not be disregarded in visual search.

Thus the dominant theory in visual search attributes the key effects to a hypothetical covert attention mechanism and ignores or discounts known visual mechanisms associated with eye movements.

Active Vision Alternative

Findlay & Gilchrist (2003) criticize the dominant theory and propose instead an *active vision* approach in which peripheral vision is used to guide eye movements that bring the high resolution portion of the retina to bear on relevant parts of the scene. The claim that the RT ms/item slopes are too fast for eye movements fails if it is possible for more than one object to be perceived, at least in peripheral vision, in a single fixation; this is consistent with the long-standing concept of the *area of conspicuity* (Engel, 1977) or *functional viewing field* (FVF, see review in Hulleman & Olivers, 2017). If the FVF is large enough, the search process can easily accommodate a fast ms/item rate even if the eyes are being moved. Accordingly, Hulleman & Olivers (2017) proposed that the ms/item characterization of visual search was a fundamental mistake, because the number of fixations, not the number of display items, accounts for visual search in these tasks. They presented a simple process model that accounted for some of the effects.

Overview

This paper starts with a reanalysis of a very high-quality dataset made available by Wolfe et al. (2010) of

performance in three classic visual search tasks, and presents an active vision model of these results based on a cognitive architecture, EPIC (Kieras, 2016; Meyer & Kieras, 1997). EPIC has components in which the visual perceptual, ocular motor, and strategy aspects of the model can be explicitly represented. The visual perceptual component captures the concept of the FVF. The oculomotor component represents the mechanisms that generate saccades with realistic timing and variability. The strategy component consists of production rules applied by the cognitive processor that decide where to move the eyes and when to respond target-present or target-absent. A manual motor component represents the time for the manual response.

While the concept of attention is clearly associated with overt behaviors such as eye movements, the concept of covert attention is generally associated with some kind of top-down direct internal control of perception by cognition. EPIC has no such mechanism. Rather, based on the available perceptual information, the strategy decides when a response can be made, or what object needs to be fixated to collect more information. If one insists on using the language of attention, then EPIC has a *very late selection* concept of attention, in contrast to the early selection posited for covert attention.

The data will first be presented, followed by a summary of the architecture and the model, and its fit to the data.

The Visual Search Experiment

The data used for this modeling was collected by Wolfe, Palmer, and Horowitz (2010) and made available for download at http://search.bwh.harvard.edu/new/data_set_files.html. Wolfe et al. focussed on the RT distributions, but this dataset is exceptional because of the well-specified stimuli and relatively large number of very well-practiced subjects. For completeness, the experimental method is re-stated here in the context of how the experiment was simulated in the model.

Method

Tasks There were three different present/absent search tasks; Figure 1 shows a sample target-present display produced by the model for each task condition, labeled in this paper as Color Single Feature (CSF), Color-Orientation Conjunction (COC), and Shape (SHP). The CSF target was a red vertical bar among green vertical distractors. The COC target was a red vertical bar among distractors that were red horizontal bars or green vertical bars. The SHP target was a "digital 2" shape among "digital 5" shapes.

Stimuli Wolfe et al. (2010) provide a good level of detail about the stimulus properties, but unfortunately, the download data set does not contain information about the actual display configuration used in each trial, so for purposes of modeling the display had to be generated for each simulated trial.

The search display was an area 22.5° × 22.5°, treated as containing 25 invisible cells of 5° × 5°; Wolfe et al. (2010) state that each object appeared in a random location within one of the cells, but did not state how overlapping objects were prevented. For the model, the random location within a cell was constrained to keep the horizontal or vertical edge of an object at least 0.25° away from the cell boundary, ensuring a minimum separation of 0.5° between adjacent objects. Set sizes were 3, 6, 12, and 18. In the model, a display was generated for each trial as follows: the set size number of distractors were first placed in randomly chosen display cells; if the trial was positive (target present), a randomly chosen distractor was replaced with a target object.

In the CSF task, the objects were 1° × 3.5° vertical bars; the target bar was red, distractor bars were green. In the SHP task, the objects were 1.5° × 2.7° character-like shapes; the target was a *2* and the distractors were *5*s. In the COC task, the objects were 1° × 3.5° bars, red or green, oriented either horizontally or vertically. The target was a red vertical bar, and distractors were red horizontal and green vertical bars. Wolfe et al. (2010) do not state exactly how COC distractors were chosen; in the model, half of the distractors were chosen to be of each type, with set size 3 special-cased so that at least one distractor of each type was present. Since a

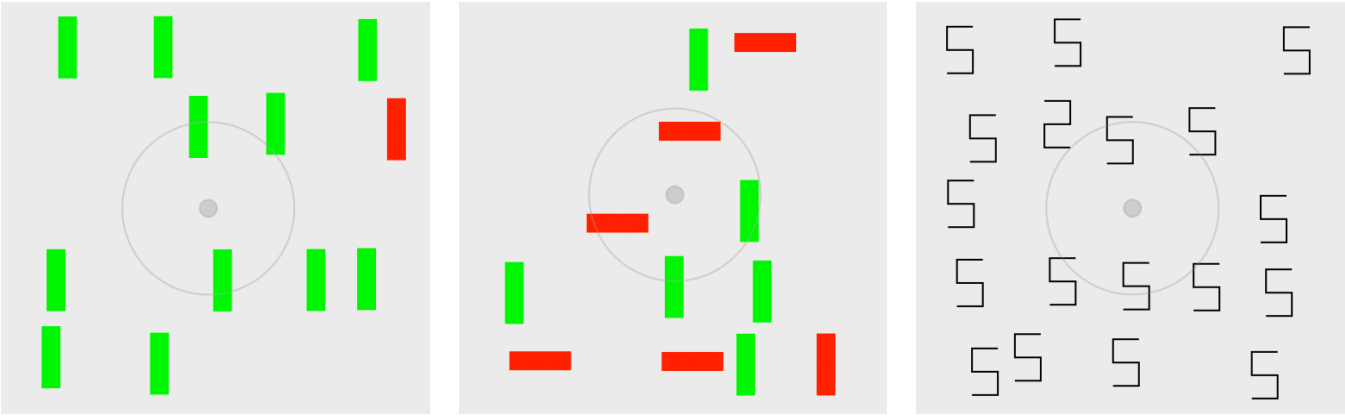


Figure 1. Sample search displays produced by the model for the color single feature (CSF), color-orientation conjunction (COC), and shape (SHP) tasks of the Wolfe et al. (2010) experiment. The concentric gray circles show the simulated eye position at the initial fixation location; for scale, the inner circle has a diameter of 1°; the outer circle is 10°.

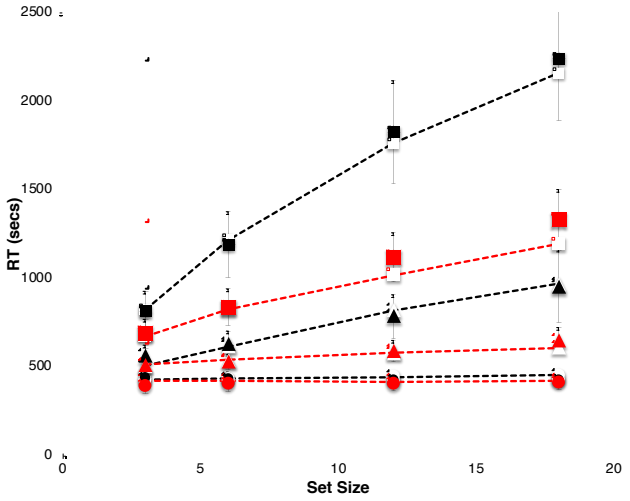


Figure 2. Observed (solid points and lines) and predicted (open points and dotted lines) for correct trial RT in each task condition. CSF: circles, COC: triangles, SHP: squares. Positive trials: red, negative trials: black. The 95% confidence intervals are based on the standard error of the mean of the subjects' mean values underlying each data point and thus reflect between-subject variability.

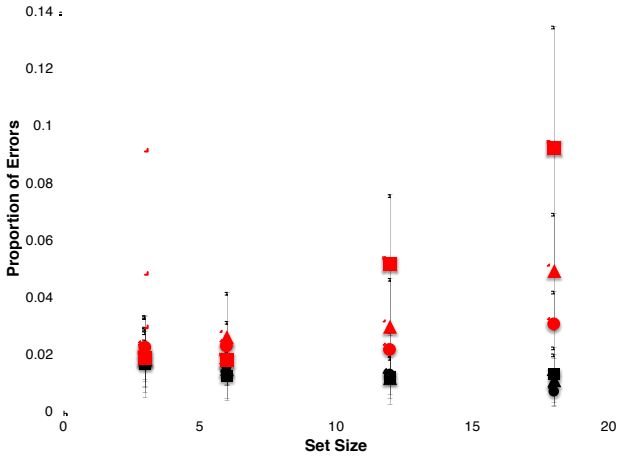


Figure 3. Observed proportion of errors in each task condition. CSF: circles, COC: triangles, SHP: squares. Positive trials: red, negative trials: black. The 95% confidence intervals are based on the standard error of the mean of the subjects' proportions underlying each data point and thus reflect between-subject variability.

positive trial display was produced by replacing a random distractor with a target, over trials, each type of distractor would appear equally often.

Design There were 10 subjects in the COC task condition and 9 in the other two. One subject was in both COC and SHP, but the data set does not identify this subject, so the task condition was treated as a purely between-subject manipulation in this paper.

Procedure Each trial began with a centered fixation cross. Subjects were instructed to "keep their eyes focussed on this cross" but because eye movements were not monitored, subjects could have moved their eyes, and based on other studies, it is likely that they did so. The search display was presented and remained visible until the subject pressed a key for target present or target absent. Subjects were instructed to respond as "quickly and accurately as possible." Correct/incorrect feedback was presented for 500 ms after each trial. Unlike many experiments, the subjects were very well practiced, with about 500 trials per subject for each combination of set size and positive/negative trial polarity.

Results

The downloaded data consisted of the RT and correct/ incorrect status for each subject in each trial at each set size and trial polarity. Following common practice in RT experiments, the data were reduced as follows: For each task condition, for each subject, the mean RT for correct trials and the proportion of errors for that subject was calculated for positive and negative trials at each set size, giving a total of 8 data points for each subject for their RT and error rate. These subject means were then averaged to produce the data points plotted in Figure 2 and Figure 3. The 95% confidence intervals around each data point were calculated by determining the standard error of that mean using the 9 or 10 individual subject means contributing to that point, thus reflecting between-subject variability, but not within-subject variability.

Wolfe et al. (2010) did not report any overall statistical tests of their results. Therefore, unequal-*N* ANOVAs were performed using the **R ez** package on the reduced data. For RT, the main effects of Task Condition, Trial Polarity, Set Size, and all two- and three-way interactions were significant ($p < .05$). For proportion of errors, whose overall average was 2.4%, the Task Condition main effect was not significant ($p > .1$) but the Trial Polarity and Set Size main effects, and all two- and three-way interactions were significant ($p < .05$). Examination of specific within-subject effects was done with Fisher Least Significant Difference values, which to avoid clutter are not shown on the graphs. But a simple summary of these comparisons is that the apparently different within-subject effects in the graphs are reliably different, even if the between-subject confidence intervals overlap.

Discussion

The RT results follow the classic pattern obtained in most visual search experiments. The RT functions are essentially flat in the CSF task (positive trial slope is about 1 ms/item), this prominent effect with the color property in a single-

feature search task is frequently described as "pop out". Otherwise, positive and negative trial RTs have a substantial slope, with the negative trial slope about twice that of the positive trials. The color-orientation conjunction task COC has a positive trial slope of about 9 ms/item and the SHP positive trial task slopes are greater at 43 ms/item. As Wolfe et al. (1989) first observed, conjunction searches can be relatively efficient, contrary to Triesman & Gelade (1980).

The error rate overall is only 2.4%, justifying the conventional approach of focussing only on the correct trial RT, but note that negative trials have a fairly constant low error rate between 1 and 2%, while positive trials produce more errors as set size increases, especially for the more difficult tasks. Overall, this rules out a speed-accuracy tradeoff effect in the RT data, but because these effects are statistically reliable in spite of large individual differences, they need to be explained in a future model. The present model applies to only the correct RT data.

Summary of the EPIC Cognitive Architecture

The EPIC architecture for human cognition and performance directly provides a general framework for simulating a human interacting with an environment to accomplish a task. The introduction of this paper provided a brief overview; see Meyer & Kieras (1997) or Kieras (2016), for a more complete description. The following summarizes the relevant components of the architecture.

In the EPIC architecture, object and their properties are formed early in vision (see Scholl, 2001). The *eye processor* component contains *acuity functions* that specify whether each visual property of each object is currently available as a function of the size of the object and its eccentricity from the current eye position. The currently available visual properties for each object are represented in the *sensory store*; the *perceptual processor* then encodes the properties of each object, possibly in relation to other objects, and passes the encoded representation on to the *perceptual store* where they are available to the cognitive processor to match the conditions of production rules. The perceptual store contains the current representation of the visual world that cognition can reason and make decisions about, including decisions about where to move the eyes next by commanding the *ocular motor processor*.

When the eyes move away from an object, the properties of the object persist for a short time (e.g. 200 ms) in the sensory store, and a long time (e.g. 4s) in the perceptual store. But if the object disappears completely, it and all of its properties will be removed from the perceptual store fairly quickly. Thus the representation persists for a considerable time as long as the scene is present; this is supported by studies summarized by Henderson & Castelhano (2005); memory for previously fixated objects was assessed in natural visual scenes, and retention times of at least several seconds were observed. The task strategy uses this retained information to avoid re-fixating an already examined object (see Kieras, 2011).

Model for Search Task RT

EPIC models for other visual search tasks are presented in Kieras (2011, 2016), Kieras & Hornof (2014), and Kieras & Marshall (2006). Constructing the model for a specific search task requires a choice of (1) visual acuity functions and parameters, (2) a parameter for the persistence time of visual properties in the perceptual store that are no longer sensorily supported, (3) a model of the "noise" in the eye movements, and (4) a set of production rules that implement the visual search strategy. Each of these will be described in the following.

Acuity Functions and Perceptual Store Retention

Despite the many decades of research on vision, the literature does not contain a comprehensive set of parametric data on perceptibility of different visual properties as a function of their eccentricity and size, so they must be estimated to fit the data. Some constraints are present in the available literature, but space limitations do not allow a review of the available data (see Findlay & Gilchrist, 2003 for an overview).

The availability of a perceptual property depends on the *eccentricity* (the distance in degrees of visual angle from the center of gaze) of the object, and on the *size* of the object (also measured in degrees of visual angle), and on the specific property involved. A separate acuity function was specified for each property of color, orientation, and shape as a Gaussian detection function that gives the probability that the property will be detected (be available) for an object with size s at eccentricity e:

$$P(\text{detection}) = P(s > N(\mu, \sigma))$$

$$\mu = a + be,\ \sigma = a \text{ constant}$$

The value μ can be interpreted as the 50% threshold for object size. The value of σ governs the steepness of the ogival detection function.

In the present model strategy, the color property is used in both the CSF and COC tasks and was constrained to have the same parameter values in these tasks; orientation was used only in COC, and shape only in SHP. The acuity parameters were determined by informal iterative fitting. The a term was held at 0.0, b was estimated as 0.1 for color, 0.25 for orientation, and 0.4 for shape. σ was held at 0.5. This corresponds to many empirical results that color is widely available, orientation less so, and detailed shape least of all. The total time for a property to appear in the perceptual store was set at 50 ms.

The availability of each property is independently resampled for all objects whenever the eyes are moved. As the eyes move around, the available properties of the same object can fluctuate, and so will not be reliably available from one fixation to the next. However, as described above, the information once acquired will remain for some time in the perceptual store, forming a stable visual representation. The retention time parameter was set at 4s, the value used in Kieras (2011) which involved a search task requiring individual object fixations.

Saccade Timing and Accuracy

The time in ms to execute a saccade of length e in degrees is provided by Carpenter's (1988) estimate as:

$$\text{saccade duration} = 21 + 2.2e$$

A variety of studies (e.g. Abrams, Meyer, & Kornblum, 1989) have shown that saccades tend to fall short of the actual fixation target, and the standard deviation of the saccade distance tends to be proportional to the distance. In the architecture, the oculomotor processor samples the length of a saccade to an object at eccentricity *e* from a Gaussian distribution:

$saccade\ length = N(\mu, \sigma))$

$\mu = g \cdot e, \sigma = s \cdot \mu$

Typical empirical values for *g* (gain) range from 0.85-0.95, and *s* (spread) is typically around 10%. In the current model, the parameters were held constant at the values suggested by Harris (1995) as optimal, namely g=0.95, s=10%. In addition, the angular direction of the saccade is also noisy, but due to the very few available studies (e.g. van Opstal & van Gisbergen, 1989) a rough estimate was used: the angle of the saccade is perturbed by a sample from $N(0, \sigma_A)$, where σ_A = 1°. Thus large eye movements often miss the object to be fixated, reducing the chances that its properties will be detected.

Task Strategy

EPIC's cognitive processor applies production rules in parallel in a 50 ms cycle. The production rules in the model are a *search-and-confirm* variation of a basic strategy used in previous EPIC visual search models. Once the display objects appear on the screen, the strategy production rules alternate between a *nomination* phase, in which rules nominate objects (possibly in peripheral vision) that are *possible targets* because a relevant property matches or is unknown, and a *choice* phase in which rules choose one of the nominated objects to move the eyes to. If multiple nominated objects match a choice rule, the closest object to the current point of fixation is chosen when the rule fires. Once the eye movement is complete, the nomination phase starts again.

Thus, over time, information about the objects accumulates until one of two choice-phase rules choose the response: If there is a nominated object which matches all of the target properties, then an eye movement is initiated to it; when complete, a second rule confirms the target properties and makes a target-present response via a manual motor processor keystroke command. If there are no nominations, this means that all objects appear to be distractors; to confirm this, if no fixation has been made yet, the strategy moves the eye to the most distant object, confirms that it is not a target, and then makes the target-absent response; unlike many models (cf. Hulleman & Olivers, 2017), there is no time-out or similar process; the lack of a possible target nomination suffices for a target-absent response.

The nomination and choice rules are very simple for the CSF and SHP tasks because only a single object property is involved. For example in the CSF condition, an object is nominated as the target if it has a red color, or as an object to be fixated if it has an unknown color. In contrast, for COC, there are three possible fixation nominations, and the strategy chooses one to fixate in the following descending priority order: Red color & unknown orientation, unknown color & vertical orientation, unknown color & unknown orientation.

RT Model Results

Using the parameter values and task strategy described above, the model was run for a total of 5000 trials in each task × polarity × set size condition; Figure 2 shows the predicted RT values. The fit is very good, with r^2=0.99, average absolute error of 32 ms, and absolute relative error of 4%. Almost all of the predicted values are within the confidence intervals; the exceptions appear in the fastest conditions, such as CSF. Examination of the model's behavior shows that the average number of eye movements (including the confirmation fixations) increases with set size and with task difficulty as indicated by RT. For example, for negative trials at set size 3, there are less than 1.1 fixations/ trial for CSF and COC, and 2.4 for SHP; for set size 18, there are fewer than 1.1 fixations per trial for CSF, 2.99 for COC, and 9.09 for SHP. So even in the easiest condition, there may be an eye movement in a trial, but the most difficult condition requires many eye movements. The flatness of CSF-like RT is commonly used to justify a special "pop out" mechanism; however, it is not needed in this model because the color property is available over a very wide area, and so a search eye movement is rarely required regardless of set size.

Accounting for Accuracy Effects

The accuracy effects, rarely addressed in the literature, need to be explained. That more errors appear in more difficult tasks only in target-present trials (miss errors) and not in target-absent trials (false alarm errors) makes sense, given that the model strategy includes a confirmation step for a target-present response. But what produces the miss errors?

An adequate model might be based on *crowding* effects in which the perception of closely spaced objects is disrupted while the primitive features might still be detected (Pelli & Tillman, 2008; Rosenholtz, 2016). The critical spacing for crowding effects is about *0.5·eccentricity*, though the magnitude of the disruption varies with the specific features involved. Apparently, crowding disrupts the association of features with their locations, and so disrupts the early-vision mechanism that integrates separate features into distinct objects. Note that in typical visual search experiments, the objects are randomly distributed in a fixed area, so set size is confounded with average object spacing. In fact, when spacing is manipulated independently, the set size effect appears to be mostly due to crowding (e.g. Wertheim et al. 2006). Rosenholtz (2016) argues that crowding effects limit peripheral vision much more than simple loss of resolution. Thus the acuity functions in the present model are apparently describing a sum of two effects of eccentricity: a strong average crowding effect and a weaker basic resolution effect.

A possible simple extension of EPIC's visual system to incorporate perceptual crowding effects would be to scramble the primitive features between objects within the critical spacing, which would tend to include more objects at larger set sizes. Such scrambling may disrupt targets and also produce illusory targets. If the strategy nominates an illusory target, the confirmation step in the present strategy

will correct the illusion, preventing a false alarm error. However, if crowding disrupts an actual target, the present strategy might conclude that it is not present and make a miss error. The effect of the scrambling would be minimal when only a single feature defines the target, as in CSF, while it would be stronger when two features must be conjoined in the same object as in COC, and even stronger when multiple primitive line-segment features have to be correctly integrated as in SHP.

Conclusion

The model built in the EPIC computational cognitive architecture provides a very accurate account of the RT data using a surprisingly simple combination of architectural components and task strategy, which together implement a basic active vision approach to visual search. Notably, there is no need for an attention mechanism of the sort proposed in the dominant theory to successfully account for the RT effects; simple perceptual mechanisms, eye movements, and a strategy that meets the task demands are all that is required. An extension of the perceptual components to include crowding effects would improve its value as a theoretical and practical tool for modeling human performance.

Acknowledgements

This work was supported by the Office of Naval Research, Cognitive Science Program under grant N00014-16-1-2560. Thanks are due to David Meyer, Greg Wakefield, and Anthony Hornof for useful discussions and comments.

References

Abrams, R.A., Meyer, D.E. & Kornblum, S. (1989). Speed and accuracy of saccadic eye movements: Characteristics of impulse variability in the oculomotor systems. *Journal of Experimental Psychology: Human Perception and Performance*, 15(3), 529-543.

Carpenter, R.H.S. (1988). *Movements of the eye*s (2nd ed). London: Pion.

Carrasco, M., & Frieder, K.S. (1996). Cortical magnification neutralizes the eccentricity effect in visual search. *Vision Research*, 37, 63-82.

Engel, F. L. (1977) Visual conspicuity, visual search and fixation tendencies of the eye. *Vision Research* 17:95–108. doi: 10.1016/0042-6989(77)90207-3.

Findlay, J.M., & Gilchrist, I.D. (2003). *Active Vision.* Oxford: Oxford University Press.

Harris, C.M. (1995). Does saccadic undershoot minimize saccadic flight-time? A Monte-Carlo study. *Vision Research*, **35**, 691-701.

Henderson, J.M. & Castelhano, M.S. (2005). Eye movements and visual memory for scenes. In G. Underwood (Ed.), *Cognitive processes in eye guidance*. New York: Oxford University Press. 213-235.

Hulleman, J. & Olivers, C.N.L. (2017). The impending demise of the item in visual search. *Behavioral & Brain Sciences*, 40(1), 1-20. doi:10.1017/S0140525X16000121, e142

Kieras, D. (2011). The persistent visual store as the locus of fixation memory in visual search tasks. *Cognitive Systems Research*, 12, 102-112.

Kieras, D.E. (2016). A summary of the EPIC Cognitive Architecture. In S. Chipman (Ed.), *The Oxford Handbook of Cognitive Science, Volume 1*. Oxford University Press. 24 pages. DOI: 10.1093/oxfordhb/9780199842193.013.003

Kieras, D.E & Hornof, A.J. (2014). Towards accurate and practical predictive models for active-vision-based visual search. In *Proceedings of CHI 2014: Human Factors in Computing Systems*. New York: ACM, Inc.

Kieras, D.E, & Marshall, S.P. (2006). Visual Availability and Fixation Memory in Modeling Visual Search using the EPIC Architecture. *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, 423-428.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review*, **104**, 3-65.

Neisser, U. (1967). Cognitive Psychology. New York: Appleton-Century-Crofts.

van Opstal, A.J, & van Gisbergen, J.A.M. (1989). Scatter in the metrics of saccades and properties of the collicular motor map. *Vision Research*, 29(9), 1183-1196.

Pelli, D.G., & Tillman, K.A. (2008). The uncrowded window of object recognition. *Nature Neuroscience*, 11(10), 1129-1135. doi:10.1038/nn.2187.

Rosenholtz, R. (2016). Capabilities and limitations of peripheral vision. *Annual Review of Vision Science,* 2, 437–57. doi: 10.1146/annurev-vision-082114-035733

Scholl, B.J. (2001). Objects and attention: the state of the art. Cognition, 80, 1-46.

Treisman, A.M, & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*, 12, 97-136.

Wertheim, A. H., Hooge, I. T. C., Krikke, K., Johnson, A. (2006). How important is lateral masking in visual search? *Experimental Brain Research*, 170, 387-402. DOI 10.1007/s00221-005-0221-9.

Wolfe, J. M. (2014). Approaches to Visual Search: Feature Integration Theory and Guided Search. In A.C. Nobre & S. Kastner (Eds), *The Oxford Handbook of Attention*. Retrieved from DOI: 10.1093/oxfordhb/9780199675111.013.002.

Wolfe, J. M., Cave, K. R., & Franzel, S. L. (1989). Guided Search: An alternative to the feature integration model for visual search. *Journal of Experimental Psychology: Human Perception and Performance*, 15, 419–433.

Wolfe, J.M., Palmer, E.M, & Horowitz, T.S. (2010). Reaction time distributions constrain models of visual search. *Vision Research*, 50, 1304-1311.

Zelinsky, G., & Sheinberg, D. (1995). Why some search tasks take longer than others: Using eye movements to redefine reaction times. In J.M. Findlay, R. Walker, & R.W. Kentridge (Eds.), *Eye movement research: Mechanisms, processes and applications*. North-Holland: Elsevier Science Publishers., 325-336.

Core High-Level Cognitive Abilities Derived from Hunter-Gatherer Shelter Building

Jerald D. Kralik (jerald.kralik@gmail.com)
Department of Bio and Brain Engineering
Korea Advanced Institute of Science and Technology (KAIST), Daejeon, 34141, South Korea

Abstract

Determining the fundamental cognitive abilities underlying human high-level cognition remains elusive. An examination of the main activities of our first *Homo sapiens* ancestors offers a normative approach. Because shelter building was critical for a nomadic hunter-gatherer and required comprehension and manipulation of the knowledge for predicting, controlling and creating, I examined shelter building. I first conducted a theoretical analysis of the necessary steps to imagine and then construct a temporary shelter in the African savanna, including the underlying cognitive abilities to do so. I then compared the results to a case study of grass-hut building by a modern-day San tribe community in Botswana, Africa. The analysis provides a set of core cognitive abilities required for shelter building, which may represent the core cognition underlying our physical intelligence. Future examination of the other primary activities of our first ancestors should help produce a complete list of our fundamental high-level cognitive abilities.

Keywords: High-level cognition; cognitive modeling; evolutionary psychology and neuroscience; anthropology.

Introduction

There have been significant advances in the study of high-level cognition — such as in attention, decision-making, reasoning, and creativity (e.g., Gazzinga, Ivry, & Mangum, 2013; Glimcher & Fehr, 2014; Helie & Sun, 2010; Holyoak & Morrison, 2012; Kralik, 2017; Kralik, Mao, Zhao, Nguyen, & Ray, 2016; Laird, Lebiere, & Rosenbloom, 2017; Russell & Norvig, 2010; Silver, Schrittwieser, Simonyan, Antonoglou, Huang, et al., 2017; Tenenbaum, Kemp, & Griffiths, 2011) — yet much remains unknown. This is especially borne out when considering current computational approaches to human cognition, with many appearing to assume that high-level cognition will arise from, for example, deep hierarchical neural networks based on associative processes. Indeed, there remains no consensus on how to model higher-level cognition, with current AI systems falling far short of human-level abilities.

What then is the best approach to identify and delineate the actual core cognitive processes underlying human high-level cognition? Researchers attempt to focus on tasks that require the highest levels of our abilities, with recent examples including video games, the game "Go", and autonomous driving (Silver, Schrittwieser, Simonyan, et al., 2017). Such approaches are valuable, but still run the risk of missing critical high-level capabilities. For example, most work concentrates on the derivation of complex action policies with a well-defined problem representation, and yet the generation of the appropriate problem representation itself is one of the most challenging problems animals (including humans) face in a potentially intractably complex real world (Kralik et al., 2016; Kralik, Shi, El-Shroa, & Ray, 2016; Sampson, Kahn, Nisenbaum, & Kralik, 2018).

Ideally, we would take a more normative approach to determine the task paradigms that best capture the fundamental nature of human high-level cognition, but to do so, it would require isolating our fundamental abilities from learning and cultural influences. For example, high-level mathematical prowess consists of both basic cognitive ability as well as substantial knowledge and skills developed from teaching and practice: i.e., from learning. How then might we isolate basic abilities from subsequent learning and cultural effects? Two possible approaches are developmental and evolutionary ones. Notwithstanding important work studying the developmental process (e.g., Tenenbaum, Kemp, & Griffiths, 2011), the approach faces limitations in that the brain is normally not fully developed, and thus developmental and learning effects accumulate. An evolutionary approach points us to biological anthropology and the evolution of *Homo sapiens*. That is, substantial evidence strongly suggests that the human brain largely finished evolving in Africa during the Pleistocene epoch (roughly 100,000 years ago), when humans were hunter-gatherers (Buss, 2015; Dunbar, 2003; Nowell & Davidson, 2010; Relethford, 2013). Thus, in principle, if we could examine the cognitive abilities underlying their main activities, we could potentially identify the core set.

To conduct such an analysis in the modern day is indeed a challenge, yet is possible via a triangulation of multiple lines of evidence: e.g., analysis of the actual Pleistocene archaeological evidence (e.g., the surviving toolkit) (Buss, 2015; Kralik, 2017; Nowell & Davidson, 2010; Relethford, 2013), logical analysis of the problem being solved, such as building a shelter using only natural elements in the local environment (e.g., Kalahari desert), and testing these analyses against modern-day examples of 'authentic' traditions passed down to current times (e.g., how to build a traditional grass hut). Regarding the latter — testing modern-day cases — given the potential diversity of solutions throughout the world, the tribes targeted for analysis can be a critical factor. Although all are valuable in helping to uncover universal abilities all humans inherited, the cleanest attempt would be to examine those most closely related to our earliest ancestors; and recent genetic evidence identifies the San/Khoisan tribes as the oldest (Schuster, Miller, Ratan, Tomsho, Giardine, et al., 2010). Of course, studying the modern San to uncover the abilities of ancestral *Homo sapiens* carries the assumption that their traditional activities reflect the original ancestral solutions (due to facing the same problem and solving it directly, logically and efficiently), but this assumption could be false. However, the assumption can

also be tested if we first conduct a theoretical analysis of how a core activity would be carried out, and then compare the conclusions to a modern-day case study.

In the current paper I take this approach by examining a core activity of the San, with the goal of identifying the basis set of the highest abilities of the human mind (prior to the accumulating effects achieved via learning processes, teaching, and culture). In the following sections, I first identify shelter building as a fundamental task to be examined. I then provide a theoretical analysis of shelter building, and then present the core cognitive abilities underlying human physical intelligence identified by this analysis. I then compare the results of this analysis to a case study of grass-hut building by a San tribe in Botswana, Africa (Pratchett, 2017). Future work that examines the other key activities of the San (such as medicinal knowledge and practices, cooking, social interactions, and fire, tool, houseware and jewelry making) will be needed to provide greater assurance that the list is complete for human higher-level cognition (Buss, 2015; Kralik, 2017; Relethford, 2013).

Why Shelter Building?

With the ultimate research aim to analyze all of the presumed major activities of ancestral humans, I nonetheless chose shelter building first for multiple reasons. Firstly, it is important to examine a task that appears most complex, and thus most taxing of our cognitive abilities. For example, it should require more than just to seek or forage and extract, with evidence suggesting these are not sufficiently complex to explain higher cognitive abilities (Dunbar, 2003). In addition, the activity ideally would be carried out by both men and women since our interest is with core universal human abilities, not more specialized cases. In fact, shelter building is intriguing not only for the apparent complexity involved, but that both sexes engage in it (and if anything, mostly by the women) (Pratchett, 2017). Finally, the activity should reflect not only understanding for prediction, approach, attainment or avoidance, but also significant *manipulation* for our own ends: i.e., controlling and creating.

These considerations nonetheless point to multiple activities, such as basic social interactions, which we have also begun examining (Lee, Kralik, & Jeong, 2018). However, an additional reason for selecting shelter building is the obvious need of our first ancestors to avoid the elements and danger as a large ape previously adapted to arboreal life in an environment turned from rainforest to woodland to savanna, with relatively little protective weaponry or terrestrial escape mechanisms: indeed, as a proverbial fish out of water. Being so exposed (with respect to heat, cold, weather, danger), natural shelters obviously had to be used, including some trees and rock formations (like caves); yet as the woodland turned to savanna these options become limited, especially if one must remain nomadic to follow food and water patterns and protect larger families. Thus, it is apparent that there was great pressure for creative solutions.

Shelter Building Analysis

To visualize a simple shelter, Figure 1 is a general illustration from the theoretical analysis, with the main elements shown.

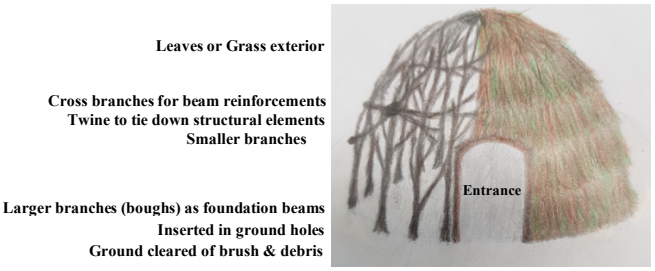


Figure 1: Shelter elements from the theoretical analysis.

The initial problem then is this: when facing elements and threats to be avoided — e.g., rain, heat, cold, predators — how would the first ancestors come to the realization that a shelter could be built? The first step is likely initial observation when in a natural shelter, like a cave or under trees (e.g., in downpour, such as during the rainy season), with the realization that it will have to be abandoned to move on to regions that may not contain such shelter (as nomadic hunter-gatherers in the African savanna). Under such a scenario, they then at some point must wonder, "couldn't we just recreate one ourselves?"

And this would require, cognitively, to construct a new problem representation with a creative, novel solution. For example, the initial action set called to mind in a downpour would be various means to seek shelter: e.g., locate a cave or sufficient tree canopy, perhaps check for safety, then move inside or under. How then could this state set (e.g., caves, tree canopies) and action set (e.g., find, enter) be expanded, ultimately to the point of creating a shelter themselves?

To focus on a specific scenario, consider seeking shelter under trees in a downpour (with results the same for other specific cases, such as with a cave). The process of coming to shelter building, then, would appear to begin with the question, "what is stopping the rain?" The answer would require an examination of the interaction between the rain and what is blocking it, i.e., the tree leaves. More specifically, it would require an examination of the contact site of collision and determining what is happening, i.e., the reason for the leaves' effectiveness, and thus the mechanism of interaction. Without the luxury of trial-and-error learning, and effectiveness at a premium, simple associative processes are not sufficient (Passingham & Wise, 2012). In other words, to be successful and efficient, our ancestors would need to identify the actual *causal relationship* — the specific mechanism of interaction — and do so in an efficient manner (reasonable amount of time, minimal errors). It thus requires *Homo sapiens the scientist*, with two critical characteristics: (1) manipulation of 'trials' as experimental tests of hypotheses to isolate the causal factors from confounds; and (2) an ability to identify the actual causal agents and their effects. Then consider each of these in turn. First, an experimental approach to problem solving would require a meta-view of the problem, i.e., manipulating the overall

problem representation and analysis of it: e.g., considering independent and dependent variables, organizing 'trials' to test their relationship, and recognizing such relationships (e.g., linear, nonlinear). Thus, ***metacognition*** is required. Moreover, such designing of experiments would also require ***mental simulation*** and ***planning***.

For the second characteristic, a proper experimental analysis would nonetheless lead to nothing visible — i.e., no perceptual features could be identified that cause the successful blocking of the rain (e.g., green color, shape). In short, it leads to a determination that there are unseen ***causal agents & forces*** (Kralik, 2017). More specifically, rain blocked by a leaf, even intuitively, would be represented as a ***collision of two counter forces driven by two underlying causal forces*** animating the rain (i.e., giving it strength and movement force) and fortifying the leaves (i.e., producing the strength and repellent properties): i.e., if R_1 = *causal relation* (as collision), and F_i = force produced by causal agent, then $R_1(F_1$(assumed causal agent$_1$, leaf), F_2(causal agent$_2$, rain)), simplified as R_1(leaf, rain) or R_1. Of course, the underlying electromagnetic forces could not be truly understood (they aren't even so now), but a proxy can be used and labeled, e.g., as a "spirit" or "essence." Indeed, all objects would now be considered as entities consisting of underlying causal agents that animate them, producing the forces that lead to their interactions, i.e., their dynamics. Thus, a kind of *animism* is anticipated not just as a religious point-of-view but as a universal cognitive construct derived from causal reasoning. Indeed, it is interesting to note that animism is thought to be universally shared by all indigenous tribes around the world (Relethford, 2013).

This theoretical understanding, then, requires the ability to ***imagine hidden causal agents*** and their ***resulting forces*** that lead to changes in objects that can be observed (Kralik, 2017). In addition, to isolate and identify these hidden causal agents, ***reductionism*** and ***inductive reasoning*** are also required. Moreover, an understanding of the relationships of causal agents from observations of specific cases (i.e., trials) requires ***abstraction*** to view (a) the specific cases from observations as an 'ordered set', and (b) the relationship between two ordered sets (e.g., a linear relationship between the independent and dependent variables).

The second major step toward shelter building is comprehending the 'roof': i.e., how the leaves are held overhead. The answer is that each leaf is held by a stem: R_2(stem, leaf); then stem held by branch: R_3(branch, stem); and, thus, branch with leaf as $R_3(R_2(R_1))$. $R_2(R_1)$, then, is a first ***nested causal relation***: i.e., ***relation of relation*** (and note a *tool* control structure).

Third, a branch with multiple stems and leaves is comprehended as $R_4(R_3$(branch, $R_2(R_1)$ * X)), with R_3 representing adherence and R_4 ***a configuration of multiple*** (i.e., X) ***nested relations***. Importantly, a key mechanism by which the mind manages these embedded relations is via ***chunking***: e.g., "R_4(etc.)" is considered a 'branch with leaves', multiple branches with leaves across adjacent trees a 'canopy', and its ability to block the rain a natural 'roof'. It is a powerful type of naming that reflects ***symbol use*** and a capacity for extensive ***cross-referencing*** (Kralik, 2017).

Fourth, one must understand how trees are held upright: $R_{7\rightarrow1}$, where R_5 = larger branch (or bough), R_6 = tree trunk (and roots), R_7 = ground (i.e., roots into ground). This extended hierarchical tree data structure reflects not only nested relations, but more clearly ***recursion*** (i.e., nested subtrees).

Fifth, now these separate tree structures (both figuratively and literally) need to be configured together to form a canopy (i.e., a more complete roof): $R_8(R_{7\rightarrow1}, R_{7\rightarrow1})$ => R_8('$R_{7\rightarrow1}$,' * X), where R_8 = circular configuration in ground of multiple (i.e., X) bough-branch-leaves structures (i.e., beams) to construct the 3D structure. R_8 is a ***relation of more clearly separate hierarchical relations***, i.e., of each tree structure or beam — a *forest* in graph-theoretic terms.

Sixth, an entrance would be needed: i.e., opening in the natural shelter under the trees to reach it (and eventually made in the hut), which is a comprehension of a new configuration of the larger one (i.e., of the larger 3D structure of the shelter) — and thus 'forests' can also be included or embedded in other relationships.

At this point the general structure of the natural shelter is understood. Thus, to actually build a shelter, this understanding is first necessary (*Homo sapiens* as scientist). But then to build it, other cognitive abilities are required — i.e., *Homo sapiens must now become the engineer*. The seventh consideration, then, is a particularly critical one. To convert the knowledge into actual construction, in brief, the key mechanism appears to be ***substitution*** of causal agents, i.e., a matching of forces and actions across different objects (i.e., via their underlying causal forces, and thus their *actions* or *forces*), and the ability to then swap causal agents (and thus various interacting objects, including self) among the roles. Indeed, the first component, matching, is essentially the comprehension of ***analogies***, i.e., matching the relations of different agents across content domains: like wind blowing branches as if carrying them. In fact, normal generalization abilities are important here but appear too limited to provide sufficient matching of actions between different causal agents (including oneself), especially across all biological and nonbiological objects of interest. Thus, ***abstraction*** is again needed to reach a level sufficient to enable matches across different objects (i.e., via their underlying causal forces, and thus their observed actions). Abstraction is reflected in concepts like hit, push, throw, drop and collide.

In addition, note that this substitution mechanism also uses explicit knowledge about *self* — e.g., labeling self as agent and modeling our own actions and relations — thus revealing ***self-reflection*** and ***self-representation*** capability (with both a type of metacognition). Moreover, when considering *action* in more detail, action selection is not merely choosing one action element from a set given the current state; rather, we need to mold actions according to affordances of a target, producing an embedded problem of 'how do I generate the necessary action (configuration)?' (Kralik, Muldrew, Gunasekaran, & Lange, 2017). In other words, we have

A_1(Self, Body), where Body = action configuration, such as that for reaching and grasping, with A_1 = actual action performed. If we substitute "Mind" for "Self", and convert A_1 to the actual underlying force it generates, F_1, the manipulation of an external object would actually be: $R_1(F_1$(Self=Mind, Body), F_2(causal agent$_2$, Target)), where R_1 = manipulation of the external object to create the desired dynamic change (producing the collision of forces between the two 'objects').

The second component of the substitution mechanism — substitution *per se*, and in particular when *self* is substituted in — is the key contact point, or pivot, between cognition and action. Indeed, it is related to the well-known mirror system discovered initially in rhesus monkeys (see Gazzinga, Ivry, & Mangum, 2013). However, in the current case it is significantly more sophisticated: for example, relating to hidden causal agents and forces, abstraction, self, and flexibility (i.e., much less "mirror"-like as a process).

In fact, the substitution mechanism appears to be a key means by which humans are able to manipulate the problem representation and problem-solving process, especially regarding adding elements to the representation (e.g., new states, stimuli, and actions). This would be expected to occur particularly during the 'scientist-experimental' periods, in which the individual is discovering new causal relationships and the causal agents underlying them (and then mapping the findings to one's one actions, often with the help of abstraction). (Another critical means to manipulate the problem-solving process is with the use of *levels* of cognitive systems, with each level having particular problem elements; Kowaguchi, Patel, Bunnell, & Kralik, 2016; Kralik, 2017; Kralik, Shi, et al., 2016; Sampson et al., 2018) Indeed, not only does the substitution mechanism provide a means by which observation of external behavior can be converted into novel actions and creative solutions, it is another example of how perception and action appear more closely aligned than typically appreciated.

Substitution and causal understanding of the underlying imagined causal agents and forces, then, enable inserting oneself (or other agents) in various places in the nested causal relations when necessary, enabling shelter construction. For example, cutting and collecting branches could occur after observing the effects of wind, and replacing it with self; or even matching one's own behavior from different circumstances, such as moving smaller branches, and then replacing the smaller with larger ones for shelter beams. This would continue at each step, e.g., configuring the beams in a circle, inserting into ground, etc. In short, the entire construction process can result once there is a proper causal understanding of each component, followed by a matching then replacement of causal agents by self or other agents.

Eighth, with a basic shelter structure in place, branches must also then be further incorporated, interleaved for reinforcement (and walls), and then more leaves are needed for covering the entire structure, including the walls. This step requires a consideration of stability and strength of the structure as a whole, and ***using relations to produce a heightened combined force***, F_x, to properly counter opposing forces (e.g., wind or gravity). Thus, R_2([$R_1(F_1$(assumed causal agent$_1$, cross branch), F_2(causal agent$_2$, beam))], F_3(causal agent$_3$, wind)).

Ninth, further reinforcement by *tying down* is needed, driving the desire to obtain a type of rope or twine, which would provide critical stability for the overall structure. In fact, this step is a clear example of additional ***subproblems***, especially for refinements of the overall structure. For example, the twine must first be prepared, such as from animal intestine, vines, or tree bark. These subproblems, then, reflect ***insertion of nested relationships***, comparable to means-end analysis (Russell & Norvig, 2010). Indeed, there are multiple subproblems (refinements), including determining the type of tree; type of external covering, i.e., leaves vs. grass (and species type); tools (such as axe, cutters), including their manufacture (Kralik, 2017; Nowell & Davidson, 2010; Relethford, 2013); and help from others, i.e., coordination of activities (Shi, Sauter, & Kralik, 2009; Shi, Sauter, Sun, Ray, & Kralik, 2010; Sun, Mao, Ray, Shi, & Kralik, 2011).

Tenth, in fact, all materials must be acquired first and the structure built in opposite order: e.g., holes being made in the ground, beams inserted into them, reinforcements made (i.e., walls), and then covering added. Thus, beginning with a need for a covering or roof as the original problem, construction yet proceeds in the opposite order, with the roof (and overall covering) produced as a final step. This again reflects mental simulation and planning, as well as — especially with respect to obtaining the needed materials — ***search, select, sort and 'data' organization processing*** abilities.

Finally, 'understanding' can provide an accurate description of what is observed (e.g., causal mechanisms of rain and tree dynamics), but to manipulate this understanding for one's own needs and capabilities (e.g., having action, size, and strength constraints of the human body), ***flexible manipulation and modifications of this knowledge*** is necessary. For example, it would not be feasible to use larger tree trunks as shelter beams, and instead, larger branches (boughs) must be acquired and inserted into the ground. Thus, further cognitive manipulation abilities are needed. Focusing on having a series of nested relations (e.g., $R_{8\rightarrow1}$), these include the ability to produce *deletions, insertions, replacements (substitutions), inversions, sections moved and rearranged, duplications*, and perhaps most notably *chaining*, which provides the transitivity necessary for another critical capacity: ***deductive reasoning***.

Core High-Level Cognitive Abilities

From the theoretical analysis of shelter building, we obtain the list of 18 core abilities shown in Table 1.

Shelter Building Case Study: San Grass Huts

Based on logic and efficiency, one might expect minimal shelters to be produced in a highly circumscribed way, as just enumerated. Nonetheless, it is important to attempt to obtain empirical evidence to test these conclusions. To do

so, I examined one case study of grass-hut building by a San tribe (i.e., the ǂ*X'ao*-||*'aen* in Botswana, Africa), in which one village member described the process as they build one in a video produced by an anthropologist who studies their lifeways in the field (Pratchett, 2017). Although it is possible that the shelter building of this modern San tribe community may be significantly different from that of our first *Homo Sapiens* ancestors, the extent to which their process matches my theoretical analysis should help to validate a basic process that would be expected to have been utilized by the original ancestors (based on the fundamental constraints of the problem, combined with a simple, logical and efficient solution). Table 2 lists the construction steps.

Table 1. The high-level cognitive abilities for shelter building.

1. **Problem solving and Decision making**
2. **Metacognition (for meta-problem-solving, i.e., manipulating the problem representation)**
3. **Mental models, Simulation, & Planning**
4. **Hidden causal agents & forces, with interaction as collision: Animism as intuitive physics**
5. **Reductionism & Inductive reasoning**
6. **Abstraction**
7. **Analogies (from matching at higher levels of abstraction)**
8. **Self-representation and Self-reflection**
9. **Substitution (of causal agents, including self, into relations)**
10. **Nested causal relations: i.e., relations of relations**
11. **Chunking, Symbolic processing, Cross-referencing**
12. **Recursion**
13. **Configurations of multiple nested relations (i.e., a 'forest'): Relation of more clearly separate hierarchical relations**
14. **Using relations to produce heightened *combined* force**
15. **Subproblems: i.e., insertion of nested relationships**
16. **Search, Select, Sort and 'Data' organization processing**
17. **Flexible manipulation and modifications of knowledge: e.g., deletions, insertions, replacements (substitutions), inversions, sections moved and rearranged, duplications**
18. **Deductive reasoning**

Theoretical and Case Study Comparison

Overall, the theoretical analysis of shelter building closely matches the case study, including material preparation, branches inserted into ground holes as beams, cross branches as reinforcements and walls, a roof, entrance, a fully covered structure (via leaves or grass) to protect from the elements, and a rope or twine to hold everything more firmly in place. In fact, further details from the case study provide even more interesting information regarding the 'refinements' or subproblems mastered, such as attention to the functional properties of specific tree and grass species, replacing the leaves with grass, the process of adding the grass, and a process used to produce twine. Nonetheless, all such additional details are readily explainable by the core cognitive abilities derived, such as replacing the leaves (observed overhead) with grass (from the ground).

The one core cognitive ability that may be less apparent in the case study is (5), i.e., actual appreciation and use of hidden causal agents and their generated force interactions among the objects (versus a simpler use of the functional properties of objects). However, multiple lines of evidence appear to reflect the deeper intuitive physics understanding. The most critical evidence found in the case study is that for *reductionism*, in which, e.g., objects are seen as a combination of parts (and the unseen 'glue' holding them together), enabling the preparation of parts as needed, including cutting branches; and then even further, the parts being seen as a combination of materials (enabling the most flexible consideration and manipulation of them). A similar assessment was derived from a detailed analysis of tool manufacturing and use from archaeological and anthropological evidence (Kralik, 2017). Put differently, we see the effects of unseen causes daily, such as the effects of gravity, and the causal mind requires a reason for them. A satisfactory explanation requires the supposition, the assumption that the causal agent must be unseen, invisible to the naked eye, which we label with an arbitrary symbol such as "spirit" or "essence" (or "force" or "gravity"). And as stated, this understanding relates to *animism* as a way of thinking, which is thought to be universally shared by all indigenous tribes, and is so by the San (Relethford, 2013).

Table 2. Actual San grass-hut construction.

1. Collection and preparation

- Gather particular species of grass as clumps (i.e., with roots), and branches of a certain tree (Za'o, Terminalia sericea) for strength and insect resistance, cutting them from trees
- Clearing for shelter entirely of sand
- Make twine (for tying into place)
  - -- From the fibers of the Za'o tree
  - -- Or from a succulent plant (!hui).
    - --- To prepare: soaked in water to soften, then beaten with pestel to expose a stringy inside, which is then twined to form the rope (for multiple purposes) (Lee Pratchett, personal communication).

2. Holes in ground, circular configuration, with digging stick, hands

3. Branches inserted in holes

- From pile of branches, inspecting for long and "bendy" ones, taking apart further if needed
- Place in holes according to size and height as "pillars"
- Fill holes to hold branches firmly
- Checking overall structure throughout

4. Roof

- Holding the branches together, bending them, and tying tightly with rope
- Further thatching with twigs and branches: i.e., adding smaller twigs and branches at the top

5. Entrance

- Wrapping smaller branches, bending and tying them (with rope) for opening
- Adding grass to "door"

6. Reinforcement, walls

- Adding more branches throughout to make the sides and top stronger ("tough and firm"), entwining them among the others (some perpendicular to beams but all directions), larger ones lower

7. Grass exterior

- First digging trench around outside to support grass at bottom, then covering with sand after inserting bottoms (i.e., the root clumps)
- Then cover the "house" by turning the grass clumps upside down, interleaving the top of the grass with the layer below
- Using stick to reach the top, "to make grass on top firm and stop the rain coming through"
- Then using larger twine to wrap around outside and tie down structure, "to avoid the wind blowing the grass away"

8. Checking and cleanup

- Checking for holes, sharp edges, sweeping debris on ground

In sum, the high theoretical and case study similarity suggests that simple shelter building yields a solution with universal principles: like a configured foundation, cross-beam reinforcements, roof,

entrance, and outer covering to protect against the elements. These basic components in turn reflect fundamental cognitive abilities required to comprehend, imagine and produce them.

Discussion

Eighteen core cognitive abilities, therefore, appear to be necessary to produce the types of shelters made by our original *Homo sapiens* ancestors, and thus constitute at least a partial list of what our minds were designed by evolution to do: in this case, to avoid a broad array of potential threats, from climate (e.g., temperature control and inclement weather) to other organisms (e.g., predators), especially at night — in the dark, and during sleep. How likely is this list complete? If we accept that an analysis of the key activities of the ancestral *Homo sapiens* should produce the core cognitive abilities, then the current findings would minimally be a necessary initial step in identifying the fundamental abilities in a way that perhaps best isolates them from possible effects of learning and culture. Of course, learning also took place among the ancestral humans. This is in fact why it is also important to attempt theoretical analyses of the problems they solved. At the same time, it is nonetheless important to assess the theoretical results with empirical ones when possible. In fact, current genetic evidence identifies San tribes as those most closely genetically related to our original ancestors (Schuster et al., 2010), and the activities from their hunter-gather lifestyle (either currently or more typically as passed down knowledge from hunter-gather times) likely reflect solutions in the Pleistocene. And yet many modern anthropologists and indigenous tribes and their advocates adamantly emphasize that their customs also reflect vast cultural evolution over time. This caveat is important and again shows why it is also critical to attempt a theoretical analysis prior to an empirical one based on any modern tribes. The fact that I obtained such a close match between the theoretical analysis and empirical description suggests that the problem itself likely produces comparable solutions (assuming a logical and efficient one).

To determine whether Table 1 is complete, other San activities that reflect ancient hunter-gather ones also need to be examined. These include fire making, cooking, houseware (e.g., pots) and tool making/manufacture, jewelry making, medicinal knowledge, and social interactions. There is indeed valuable information on most of these, but a detailed examination of the necessary underlying cognitive abilities is needed. An additional critical question that such future findings should shed light on is the extent the mind/brain is organized around general versus specialized content domains. A direct comparison of the generated lists of necessary cognitive abilities should provide some clarity.

In considering the novelty of the current findings, it is of course true that all items in Table 1 are well known and actively studied (e.g., Gazzinga, Ivry, & Mangum, 2013; Glimcher & Fehr, 2014; Helie & Sun, 2010; Holyoak & Morrison, 2012; Kowaguchi et al., 2016; Kralik, 2017; Kralik, Shi, et al., 2016; Kralik, Mao, Zhao, Nguyen, & Ray, 2016; Laird, Lebiere, & Rosenbloom, 2017; Russell & Norvig, 2010; Sampson et al., 2018; Silver, Schrittwieser, Simonyan, Antonoglou, Huang, et al., 2017; Tenenbaum, Kemp, & Griffiths, 2011). However, to my knowledge, no current AI system is explicitly based on these 18 core abilities. For example, most systems build knowledge structures around objects, more than around the underlying causal agents animating the objects (although see Battaglia, Hamrick, & Tenenbaum, 2013). Of course, the functional properties of objects are utilized, but the argument here is that they have to this point perhaps been too subordinated to basic object knowledge. Exactly what this means in terms of applicability will require a computational implementation of such a system (i.e., based on Table 1), which I am currently undertaking. Indeed, a computational implementation will also help determine whether such core cognitive abilities are actually sufficient to then simulate, together with learning and cultural knowledge, more of our higher abilities and achievements (such as an understanding of higher math, and from building shelters to spaceships).

Finally, a major thrust of this research program is an attempt to better understand how problem representations themselves are generated and modified (Kralik, Mao, et al., 2016; Kralik, Shi, et al., 2016; Sampson et al., 2018). This critical problem appears to be understudied, and yet one that appears to have been enduringly confronted by our ancestors: 'how do I solve a problem that I never faced before, being a fish-out-of-water as a large and relatively defenseless ape adapted for arboreal life?' In any event, it is believed that the approach advocated here should complement others with the aim to uncover and eventually simulate the highest abilities of the human mind and brain.

References

Battaglia, P. W., Hamrick, J. B., & Tenenbaum, J. B. (2013). Simulation as an engine of physical scene understanding. *PNAS*, 110(45), 18327–18332.

Buss, D. (2015). Evolutionary Psychology. New York: Routledge.

Dunbar, R. I. M. (2003). The Social Brain: Mind, Language, and Society in Evolutionary Perspective. *Ann. Review of Anthropology*, 32(1), 163-181.

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.

Hélie, S. & Sun, R. (2010). Incubation, Insight, and Creative Problem Solving: A Unified Theory and a Connectionist Model. *Psychological Review*, 117, 994-1024.

Holyoak, K. J., & Morrison, R. G. (Eds.). (2012). The Oxford Handbook of Thinking and Reasoning. Oxford University Press.

Kowaguchi, M., Patel, N. P., Bunnell, M. E., and Kralik, J. D. (2016). Competitive control of cognition in rhesus monkeys. Cognition, 157: 146-155.

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., Mao, T., Zhao, C., Nguyen, H. T., and Ray, L. E. (2016). Modeling incubation and restructuring for creative problem solving in robots. *Robotics and Autonomous Systems, Special Issue on Robotics and Creativity*, 86: 162-173.

Kralik, J. D., Muldrew, D. B. C., Gunasekaran, D., and Lange, R. D. (2017). Cognitive control for goal-directed reaching in a humanoid robot. *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*.

Kralik, J. D., Shi, D., El-Shroa, O. A., and Ray, L. E. (2016). From low to high cognition: A multi-level model of behavioral control in the primate brain. *Proceedings of the Annual Meeting of the Cog. Sci. Society.*

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.

Lee, J., Kralik, J. D.*, and Jeong, J.* (2018). A Sociocognitive-Neuroeconomic Model of Social Information Communication: To Speak Directly or To Gossip. *Proceedings of the Annual Meeting of the Cog. Sci. Society.* *Co-corresponding authors.

Nowell, A., & Davidson, I. (2010). Stone Tools and the Evolution of Human Cognition. Boulder: University Press of Colorado.

Passingham, D., & Wise, S. P. (2012). The Neurobiology of the Prefrontal Cortex. Oxford: Oxford University Press.

Pratchett, Lee J. (2017). Community-based digital documentation of Ju|'hoan and ǂX'ao-||'aen: audio, video and text archives of language and culture diversity. ID: M!a gu tju - let's build a house. London: SOAS, Endangered Languages Archive, ELAR. URL: https://elar.soas.ac.uk/Collection/MPI854174 (accessed Feb. 16, 2018).

Relethford, J. H. (2013). The Human Species. NYC: McGraw-Hill.

Russell, S., & Norvig, P. (2010). Artificial Intelligence. Upper Saddle River, NJ: Prentice Hall.

Sampson, W. W., Khan, S. A., Nisenbaum, E. J., and Kralik, J. D. (2018). Abstraction promotes creative problem-solving in rhesus monkeys. Cognition, 176: 53–64.

Schuster, S. C., Miller, W., Ratan, A., Tomsho, L. P., Giardine, B., Kasson, L. R., et al. (2010). Complete Khoisan and Bantu genomes from southern Africa. *Nature*, 463(7283), 943–947.

Shi, D., Sauter, M. Z., and Kralik, J. D. (2009). Distributed, Heterogeneous, Multi-Agent Social Coordination via Reinforcement Learning. *Proc. of the IEEE Int. Conference on Robotics and Biomimetics (ROBIO)*.

Shi, D., Sauter, M. Z., Sun, X., Ray, L. E., and Kralik, J. D. (2010). An extension of Bayesian game approximation to partially observable stochastic games with competition and cooperation. *Proceedings of the International Conference on Artificial Intelligence (ICAI)*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., et al. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676), 354–359.

Sun, X., Mao, T., Ray, L. E., Shi, D., and Kralik, J. D. (2011). Hierarchical state-abstracted and socially-augmented Q-learning for reducing complexity in agent-based learning. *Journal of Control Theory and Applications*, 9: 440-450.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., & Goodman, N. D. (2011). How to grow a mind: Statistics, Structure, and Abstraction. *Science*, 331, 1279-1285.

A cognitive model of switching between reflective and reactive decision making in the Wason task

Othalia Larue (othalia.larue@wright.edu), Alexander Hough (hough.15@wright.edu), Ion Juvina (ion.juvina@wright.edu)

Department of Psychology, Wright State University, ASTECCA laboratory
Dayton, OH 45402 USA

Abstract

Using the Wason card selection task, we model how humans decide to engage in further deliberation after generating an intuitive response. Central to our model is the Feeling of Rightness which is the fluency with which one makes a decision, and how good one feels about it. The model is implemented in a cognitive architecture, ACT-R. A core affect mechanism and the Feeling of Rightness component are used to drive the decision. A training procedure was used to simulate individual differences in heuristic and analytical behavior. Different degrees of reinforcement were then used to allow the model to learn logical skills, and acquire differences in FOR, which determined the degree of further deliberation. By relying on ACT-R's different memory components and the core affect mechanism we were able to reproduce the variation between analytical and reactive behavior.

Keywords: cognitive decoupling, cognitive architecture, Wason task, deliberation, Feeling of Rightness, heuristic.

Introduction

The abstract Wason card selection task (Wason, 1968) is a reasoning task that involves two types of answers, which reflect different types of processing. In this context, fast (heuristic) processing results in an incorrect answer, while slower and more analytical processing is more likely to result in a correct answer.

In previous work (Larue et al, 2013), we explored this phenomenon by implementing a specific theory (the tripartite framework, Stanovich, 2009) in a computational model. Stanovich's tripartite framework provides an explanation of how reflective and reactive human behavior emerges from the interaction of three distinct cognitive levels (or minds): autonomous, algorithmic, and reflective. We showed how the three levels contribute to the different types of answers in the Wason card selection task. However, we did not explain what triggered each of the three levels to become involved in the decision process. Here, we will do so using the same task.

We will briefly review the extant literature about human performance on the Wason card selection task, what functions are involved, and how they are triggered to improve decision-making. For the latter we will specifically go over the notion of Feeling of Rightness (FOR; Thompson, Evans, & Campbell, 2013), which describes how good we feel about the answer we produced and helps us decide if we want to engage in further deliberation. Then, we will present a computational cognitive model that attempts to explain how humans decide to engage in deliberation in the Wason task by integrating both Stanovich's tripartite framework (Stanovich, 2009) and a complementary notion, the FOR (Thompson, Evans, & Campbell, 2013) into the ACT-R cognitive architecture (Anderson, 2007). Finally, we will present qualitative results of our computational model on the Wason task.

Related work

In the abstract Wason card selection task (Wason, 1968), four cards with a letter on one side and a number on the other side are presented. The visible sides are "A", "D", "3" and "7". The following rule is also presented: "If there is an A on one side of the card, then there is a 3 on the other side of the card". Then, the participants are asked: "Which card(s) should you turn over in order to test the truth of the rule ?"

The rules of inference that should be used for deliberation are: *modus ponens* (if P then Q, P: therefore, Q) and *modus tollens* (If P then Q. Not Q: therefore, not P). However, these rules do not seem to be systematically applied by participants as evidenced by the variety of answers: ("A" and "3"), ("A"), ("A" and "7"). The most frequent answers are "A" and ("A" and "3").

Different theories have attempted to explain these results, such as the mental model theory (Johnson-Laird, 1980) and the heuristic analytic theory (Evans, 2006), but have not provided a complete explanation for the variety of observed behaviors. Stanovich's tripartite framework (Stanovich, 2009) is a dual process theory (i.e., a theory distinguishing two processing styles: type 1, that is fast and automatic, and type 2 that is slow and deliberative) that fills this gap. It provides an explanation of how both reflective (i.e., type 2 processing characterized by sequential processing) and adaptive (i.e., type 1 processing characterized by reactivity) behaviors emerge from the interaction of three distinct levels: (1) an autonomous mind responsible for fast context-sensitive behaviors (type 1), (2) an algorithmic mind responsible for control (type 2), and (3) a reflective mind responsible for deliberative processing (type 2). Engaging in deliberation involves a process called cognitive decoupling. It is launched by the reflective level and carried out at the algorithmic level. Cognitive decoupling is an off-line simulation: modules fitting its description can be found in the psychology research (Leslie, 1987) and in cognitive architectures (Larue, Poirier, & Nkambou, 2013; Sun, 2015).

The tripartite theory explains three different behaviors observed in the Wason task: (1) a complete reliance on the autonomous mind leads to the frequently observed error (i.e., selecting two cards ("A" and "3"), one with the correct

answer ("A") and one with the incorrect answer that are found in the wording of the task), (2) a partial deliberation leads to selecting only one of the correct answers (*modus ponens*), the one containing the antecedent, and (3) a complete deliberation allows for the activation of the counter-information rule (*modus tollens*) which leads to a response that is less frequently observed ("A" and "7").

The initial heuristic judgment ("A" and "3") is cued and automatic (matching to the wording). Why is this judgment accepted without more analysis? According to Thompson et al. (2013), we are tempted to accept this initial response as our final one because it is compelling; it has the feeling of rightness (FOR). It is a metacognitive experience that determines if one proceeds to further analysis; it is essentially the perceived confidence that an initial answer is correct.

In previous work (Juvina, Larue, & Hough, 2017), we showed that an affective modulation of memory helps to make better decisions in complex tasks that exceed human limited memory capacities. In this paper, we will use the same mechanism to explain people's judgment of their initial answer based on its fluency. We hypothesized that the fluency of the initial answer creates the gut feeling that leads to being confident in this first answer. In our model, we assume that FOR is the component deciding if one has more of an analytical profile or a reactive profile. It reflects how fast it was to produce a first non-analytical "gut" answer and influences the extent of subsequent processing.

Model

The model[1] is implemented in ACT-R (Adaptive Control of Thought – Rational; Anderson, 2007), a unified theory of human cognition used to develop computational models of various cognitive tasks. ACT-R is composed of various modules: goal, imaginal, visual, aural, manual, vocal, and two memory modules: declarative memory (facts) and procedural memory (know-how). It is a hybrid cognitive architecture. Symbolic components are combined with subsymbolic components: the retrieval of a fact from declarative memory depends on subsymbolic retrieval equations (pondering the context and history of retrieval of the fact), and, the selection of a rule depends on a subsymbolic quantity called utility, which computes costs and benefits associated with the rule. Learning processes act at both subsymbolic and symbolic levels. Additionally, we use the core affect mechanism (Juvina, Larue & Hough, 2017): the memory elements (chunks) are reinforced through reward patterns that occur within the environment.

Model elements

Memory representation of the "mental models" Chunks are elements of the declarative memory in ACT-R. As shown in Figure 1, the representation of a mental model includes two rule structure chunks and four card chunks. The first line of Figure 1 means that the chunk with the name rule1 is a chunk of the type rule with a slot "if" that has the value "p", a slot "then" with the value "q", the slot "notif" with the value 0 (no negation), the slot "notthen" with a value 0 (no negation). This chunk represents the logical structure of modus ponens and will be used by the procedure modus ponens. The third line of Figure 1 means that the chunk with the name card1 is a chunk of the type card with a slot "letter" that has the value "A", a slot "meaning" with an empty value (""), the slot "not" with a value 0 (no negation), no number (nil), and the slot "deco" with a value 0 (to signify that it is not part of the decoupling inner simulation). As the model processes information, the card1, card2, card3, card4 chunks will be modified.

```
(rule1 isa rule if "p" then "q" notif 0 notthen 0)
(rule2 isa rule if "q" then "p" notif 1 notthen 1)

(card1 isa card letter "A" meaning "" not 0 number nil deco 0)
(card2 isa card letter "D" meaning "" not 0 number nil deco 0)
(card3 isa card number "3" meaning "" not 0 letter nil deco 0)
(card4 isa card number "7" meaning "" not 0 letter nil deco 0)
```

Figure 1: Memory representations in ACT-R.

When the decoupling starts, the model's inner simulation is populated by the card elements that could be retrieved at that time ('card1', 'card2', 'card3', 'card4' in Figure 1). To mark the element in declarative memory as part of the decoupling operation, its "deco" slot is switched to 1. *Modus ponens* and *modus tollens* rules are only applied on an element with a "deco" slot of 1. When decoupling ends, the "deco" slot is switched to 0.

When the *modus ponens* rule (which uses the *modus ponens* structure chunk) is applied, it modifies the "meaning" slot as "p" and 'card2' "meaning" slot as "q". The *modus tollens* rule will lead to the modification of card4 meaning slot as "notQ".

The resulting mental representation in declarative memory is the mental model of the task at a given time. The merging with declarative memory that occurs at the end of the cognitive decoupling reinforces the chosen mental model in memory.

How FOR is modeled? Cards are represented as chunks in the declarative memory (see Figure 1). Rethinking times, answer changes, and fluency are a function of the value of FOR. FOR depends on the time required to achieve the retrieval of "A" and "3" through an initial priming rule: the higher the FOR, the less you engage in deliberation. The FOR-inverse variable is a measure of how fast the model retrieved those two first cards/answers, as measured by the temporal module. FOR-inverse acts as a gateway for further processing when it is above a threshold. Thus, the time required to complete the initial retrievals is assigned to the FOR-inverse variable; when the value of this variable is below threshold, the model goes with the initial answer (i.e., type 1 processing); when FOR-inverse is above threshold, cognitive decoupling is launched and the model engages in further processing (i.e., type 2 processing). Every reward sent in the system is a function of FOR. including the reward sent

[1] Model code available: thttp://psych-scholar.wright.edu/astecca/

at the end of decoupling (see analytical thinking column of Figure 2): it will determine the extent to which a decoupling result (i.e., wrong, partial, and complete) is taken into account during answer selection in our model.

Answer selection The model produces an answer when the valuation of a representation is above a certain threshold. The valuation and arousal values, which help to define the core affect, are sub-symbolic quantities added to the current sub-symbolic equations of ACT-R. It allows for a specification of how emotion meshes with other cognitive processes. A detailed explanation of the core affect model is out of the scope of this paper and can be found in Juvina et al. (2017).

Core affect augments ACT-R's general equation.

The augmented core-affect activation equation is:

$$A_i = B_i + S_i + P_i + V_i + Ar_i + \varepsilon_i \qquad (1)$$

As part of the general activation equation, A_i is the activation of the chunk i, Bi the base-level term reflecting its recency and frequency, Si is the spreading term (context effect) and Pi is the partial matching term. This general equation is augmented with the valuation which is learned from rewards. Vi is the valuation term and reflects the rewards received by the model after referencing chunk i. Ari is the arousal term which reflects the importance of chunk i and is computed as the absolute magnitude of the valuation term.

$$V_i(n) = V_i(n\text{-}1) + \alpha v[R_i(n) - V_i(n\text{-}1)] \qquad (2)$$

Vi(n) is the valuation of chunk i after its nth update. Vi(n-1) is the valuation of chunk i prior to its nth update. αv is the learning rate for valuations. Ri(n) is the effective reward value received by chunk i before its nth valuation update.

When a reward is triggered, valuations are updated. All chunks that have been referenced within a time window are updated. The time window is controlled by the parameter :vtw (valuation time window).

Rewards are a function of the initial FOR (negative factor in the case of negative reward). It will affect the answer selection ("yes" or "no" answers will be produced according to how the model "feels" about the answer). A reward is sent after the initial priming so that the priming of the first answer is salient. When the model engages in cognitive decoupling, a negative reward attenuates the priming of that first answer (but does not erase it). Rewards also reinforce the chunks of the cards retrieved from memory during *modus ponens* and *modus tollens*.

Rules involved in cognitive decoupling

These rules will be used if the FOR-inverse is above a certain threshold (Parameters of the model can be seen in Table 1). Successive retrievals of chunks of the type "card" populate the representations used by cognitive decoupling. Their "deco" slot is modified from 0 to 1. Rules are applied on chunks from the declarative memory which "deco" slot has been marked as 1.

Modus ponens Before *modus ponens* is applied, a procedure inhibits the previous answer (from priming) by triggering a reward, which will decrease its valuation. This procedure is applied during the cognitive decoupling. Through an exploratory procedure, "A" is identified as P (the meaning slot of the chunk with the letter slot with value "A" is filled with "p"). *Modus ponens* only retrieves the "A", which will be reinforced by the retrieval and the positive reward sent after the end of the decoupling, leading the model's instance to answer "A". If the *modus ponens* rule structure can't be retrieved, *modus ponens* can't be applied, resulting in a cognitive decoupling with a wrong answer ("A" and "3"). The chunks with the "A" and "3" slot values are inhibited at the beginning of decoupling; however, the positive reward sent at the end of the cognitive decoupling will still reinforce the "A" and "3" who were the last element retrieved before decoupling (as they are still in the valuation window).

Modus tollens This rule is applied during the cognitive decoupling after the *modus ponens* rule. Before this rule can be applied, an exploratory rule is applied; "7" is identified as notQ (the meaning slot of the chunk with the number slot with value "7" is filled with "notQ"). If declarative memory retrieval requests fail during the exploratory rule, *modus tollens* can't be applied. As the chunk with letter slot "A" and number slot "7" are retrieved by the exploratory rule, they are reinforced by the retrieval and the positive reward sent after the end of the decoupling, leading to answer "A" and "7".

Temporal module ACT-R has a temporal cognition module (Taatgen, Rijn, & Anderson, 2007), allowing it to account for how one estimates time that passed. If the model has access to the temporal module, it can estimate how much time has passed. In the "fast" condition, this value is factored in when the model evaluates its current FOR. As more time passes, the FOR-inverse variable increases and FOR decreases. In the fast condition, to account for time pressure, a factored value of this time is added to the FOR-inverse-threshold that will lead to a deeper analysis of the problem to be done. This is what leads instances to stop "thinking" more often in this condition.

Table 1. Training and model parameters

Parameter	Value
Baseline heuristic behavior training iterations	3
Additional training iterations	1
Proportion of formal logic in additional training	20%
Proportion of heuristic in additional training	80%
:rt	-2.4
For-inverse-threshold	15
:vw	0.8

Training procedure

A training procedure allowed us to create individual differences in our simulation: differences in personal logic and impersonal logic.

Personal logic: creation of the "if" heuristic First, as a baseline the heuristic behavior (also referred to as personal logic) in each individual instance of the model is reinforced. Consistent with Evans (1998) and Stanovich (2009), the "if" heuristic directs the participant's attention to the "if" statement and leads to an initial bias in the building of a mental representation of a task (where both elements are identified as true). In order to represent this heuristic, we placed a structural representation of the "if" statement in declarative memory, which is retrieved for application in a rule (matches-if-form) and applied by another rule (matching).

To simulate the frequent use of the "if" heuristic we make the model fire the "if-form" rule repeatedly leading to an increased utility of this rule. Additionally, we used production compilation (Taatgen and Lee, 2003); a mechanism to model skill acquisition in ACT-R. Because the two rules retrieving the two elements of the "if" statement ("A" and "3") fire in sequence and often, the production compilation mechanism creates a new rule out of those two rules.

Furthermore, through the production compilation mechanism, if one of the rules retrieved an element from memory (here the "if" structural statement), that element is directly integrated into the new rule. Therefore, according to the degree to which our model is trained, it is more sensitive (answers faster) to "if" statements. The formed new rule will initially have a low utility but through training will increase its utility and be selected more often (and differently in each instance of the model) representing how much the rule is expected to contribute to the model's goal.

Impersonal logic: advanced logical aptitudes After initial training, an additional training is administered, which simulates individual differences. This training reflects the lesser proportion of people who received a strong formal logic training (impersonal logic). This is accomplished by training 20% of the simulated population with formal logic, while the other 80% received additional personal logic training.

To represent that participants are naturally more exposed and trained to apply the *modus ponens* (compared to the *modus tollens*), the training procedure involved more retrieval of the *modus ponens* representation in declarative memory than for the *modus tollens*; leading to their reinforcement in declarative memory. Differences in chunk retrieval create varying activations between the chunks and modulates if and how fast they are retrieved. Failure in the retrieval of the *modus tollens* representation is more frequent, leading to its less frequent application.

Additionally, the rules involved in the analytical processing in the model were activated, which increased their utility in procedural memory. Since the production compilation was active for the whole training procedure, in training where the *modus tollens* rule was correctly retrieved, the ending rule of the *modus ponens* is combined with the beginning of the application of the *modus tollens* guaranteeing the application of *modus tollens* after *modus ponens*. Without rule compilation, the transition between *modus ponens* and *modus tollens* would not be automatic. In those model instances, the application of *modus tollens* after *modus ponens* was automatic since the production compilation mechanism has eliminated one retrieval operation. The frequent retrieval of the *modus tollens* representation, followed eventually by its application and the compilation of rules, is how some simulated participants developed advanced logical skills.

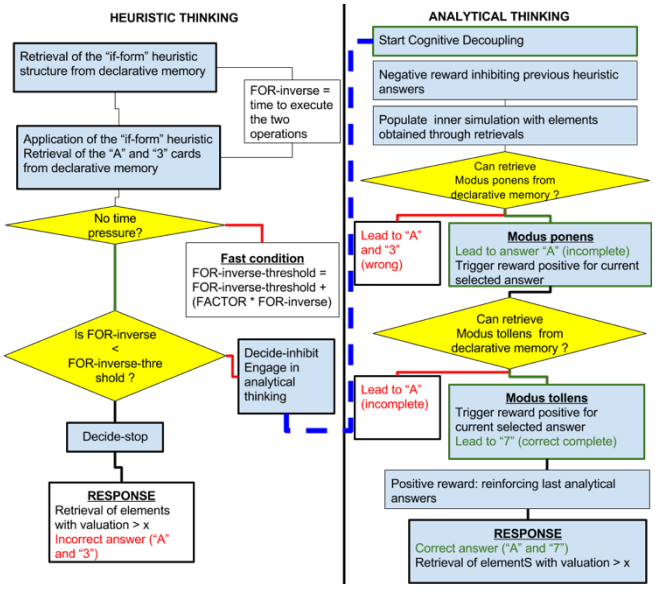


Figure 2: Model dynamics

Model dynamics

We describe here the processing in a model instance where the model starts out thinking heuristically and moves towards analytical thinking, depending on the FOR. (this is further explained in the following sections). In the declarative memory of the model, the instruction "if 'A' then '3'" is present. A first rule will retrieve the "if-form" from declarative memory (shown in the heuristic thinking column of Figure 2). Another rule will match the retrieved if-form to the instruction. This leads to the retrieval of card "A" and "3" representations. The retrieval of the two cards leads to their reinforcement in declarative memory in ACT-R. At this point, time pressure and the value of FOR-inverse variable are calculated. In the fast condition, a factored value of this time is added to the FOR-inverse-threshold to account for time pressure. Depending on time pressure and the value of FOR-inverse, two rules can then fire: decide-inhibit or decide-stop. If the variable is under the FOR-inverse-threshold (the answer was fluent) then the model's instance will stop and provide an answer (i.e., decide-stop). Otherwise, the instance will engage in analytical thinking to further analyze the problem before it answers (shown in the right column, transition between the left column and the right column is shown by the dashed blue line in Figure 2). Cognitive decoupling starts with a negative reward inhibiting

the previous answers activated at the heuristic phase (retrieved cards "A" and "3"). Then a retrieval of 'card' chunk types (see Figure 1) provides the elements to populate the inner-simulation of the problem. Note that, if the elements in declarative memory are under a certain retrieval threshold they won't be included in the inner-simulation. This is the focal bias of the model; if the elements that would lead to an activation of *modus tollens* (here the "7" card) are not in the inner simulation then this rule can't be applied. When elements (chunks) are retrieved for the inner simulation, their deco (decoupling) slot is changed from 0 to 1.

The *modus ponens* rule is the first one that the instance will try to apply by retrieving it from memory, if this retrieval fails, the decoupling will end and the first answer will be left unchanged (i.e., "A" and "3"). If the retrieval succeeds, it leads to the incomplete answer of "A", which receives a positive reward. A rule then tries to match the structure of *modus ponens* to the elements of the inner simulation. If the rule does not match, it won't fire and *modus ponens* won't be applied. The *modus tollens* rule is the second one to be applied. The *modus tollens* structure is retrieved from memory, but if this retrieval fails, the only answer produced comes from the application of *modus ponens* (The incomplete answer of "A"). If the retrieval succeeds, a rule will try to match the structure of *modus tollens* to the elements of the inner simulation. Again, if the rule does not match, it won't fire and *modus ponens* won't be applied. At the end of cognitive decoupling, the elements retrieved by the both the *modus ponens* and *modus tollens* rules are placed sequentially in the imaginal module and their deco (decoupling) slot is changed back from 1 to 0, and reentered in declarative memory (reentrance in declarative memory leads to reinforcement). Additionally, a positive reward is produced, reinforcing the most recently retrieved elements. The final answer of the system is produced based on the valuation of each card: each card is retrieved and if it is under a valuation threshold the instance answers "no" for this card, otherwise it answers "yes". The operation is repeated for each card.

Results

Experimental procedure

The experimental procedure is similar to the one described in Thompson et al. (2013). We qualitatively compared the results of our computational models to the results obtained in Thompson et al. (2013). The simulation consisted of 100 trials (100 model "participants" with 1 trial for each model "participant"). For each trial, the model was run twice: once in the fast condition and once in the complete condition (without time pressure). At the beginning of each trial, the training procedure was implemented to create individual differences between subjects. Each subject received a strong training in what we called "personal logic" (making deduction from the wording of the task according to linguistic cues). Subjects received varying training (2/10 ratio) in impersonal logic (logic rules *modus ponens* and *modus tollens*). The model was reset between each trial, after completion of both the fast and complete conditions, to keep the same individual profile of that model in both conditions and transfer resulting activations from the previous fast condition into the complete condition.

Response times and answer changes

The FOR shown on the graph was computed by inversing FOR-inverse and dividing its value between the maximum FOR-inverse and the minimum FOR-inverse by 8 (assigning FOR from 1 to 8 depending on which interval the FOR-inverse belonged to). Human participants in the study of Thompson et al. (2013) provided a Likert scale evaluation of their FOR. In our model, we use the internal value of the model's FOR (computed with the time it takes the model to finish applying the first "matching" heuristic).

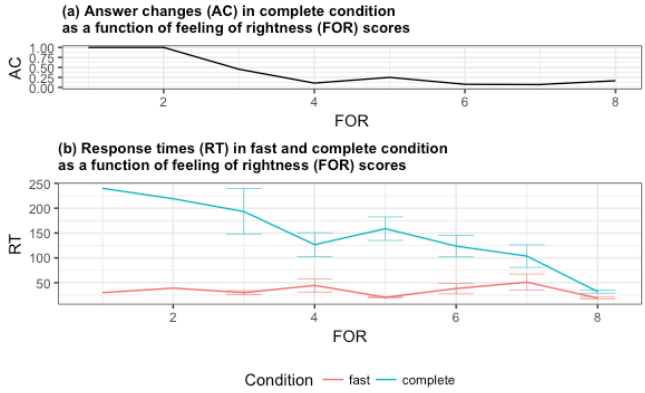


Figure 3: (a) Answer changes (AC) as a function of FOR. (b) Response Times (RT) as a function of FOR

Figure 3 presents the Response Times and answer changes according to the FOR grouped in seven intervals. Note that, while Thompson's study is similar to ours, it included a variety of wordings of the task we did not reproduce here. We were able to produce results qualitatively similar to the original study (see Figure 4 for original study's results). The second response times in the slow condition and the complete condition correlate with the answer changes indicating that answer changes result when further analytical processing occurs. There also is less individual variability in the response times as we reach the extreme values of the FOR (higher and lower). There is a significant difference in response times between the two conditions ($t(137.54)=7.32$, $p < 0.001$). Response time for higher FOR in the complete condition are also closer to values of the fast condition; indicating that the model adequately reproduced participants tendency to not engage in analytical processing when they felt right about their first answer.

The standard error observed in Figure 3.b comes from the variation between individuals we introduced to training. There is no significant difference between the two conditions when FOR is higher than six. The instances where the model was trained more in personal logic are the ones with the highest FOR as the retrieval of the "if-form" structure and the application of the if-form rule (after production compilation) goes faster. Even when offered a chance to reconsider their answer (with a For-inverse-threshold that is not affected by

the time pressure factor), the model's instances keep their first answer. Training in personal logic was provided to a higher proportion of simulated participants (models) to account for "real-life" experience. Therefore, there are more instances with a higher FOR (e.g. more heuristic thinking). Furthermore, fewer instances received a higher training in personal logic and even if they received the training, they still have more chances to not complete a cognitive decoupling (inner simulation populated with an incomplete representation of the problem, *modus ponens* structure not retrieved, *modus tollens* structure not retrieved, *modus ponens* or tollens can't be applied because they don't match with any element from the incompletely populated inner simulation). The difference between the two conditions is significant from FOR values less than 6 (t (107.36)=-7.20, $p < 0.001$).

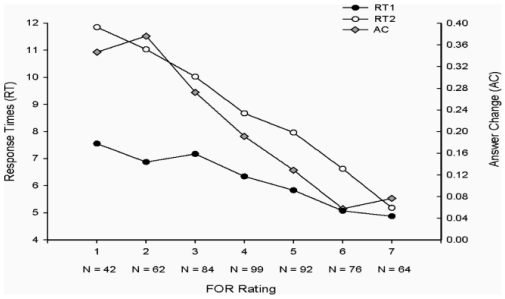


Figure 4. RT1 (first response time), RT2 (second response time), AC (Answer change) from Thompson et al. (2013)

We were able to reproduce the variation tendency of response times. However, our response times were not at the same time scale as Thompson's study (see Figure 4): our model did not parse the instructions like a human participant would. The purpose of this paper was not to reproduce the time participants spent reading the task material and processing the visual aspects of task (looking at the card). Here, we were mainly concerned with the processing that occurred after the instructions were read and participants processed the visual presentation of the cards.

Different types of reasoning

Figure 2 presents the different types of reasoning the system engages in to produce the "A" and "3" (incorrect answer), "A" (correct but incomplete answer – *modus ponens*), "A" and "7" (complete correct answer – *modus ponens* + *modus tollens*). Figure 2 shows the model can reproduce those three types of reasoning that correspond to different level of involvements of analytical processing. "A" and "3" is produced when no cognitive decoupling happens. *Modus ponens* and *modus tollens* answers happen with cognitive decoupling. If only *modus ponens* is applied however, it means that the cognitive decoupling was incomplete. The *modus ponens* and *modus tollens* rule structures are present in the declarative memory of the system, depending on the activation noise set in the system. An instance of the model might fail to retrieve a rule while another instance of the model won't. This results in individual differences in reasoning for each instance of the model.

Conclusion

In humans, the tendency to engage in analytical thinking and cognitive decoupling varies individually, and in this paper, we focused on reproducing the mechanism that determines if one will or will not engage in further analytical processing. In our model, we assume that FOR is the component necessary to decide if one will have more of an analytical profile or a reactive profile. The FOR is similar to the fluency of the model, how fast it was able to produce a first non-analytical "gut" answer, which influenced the extent of subsequent processing.

While we were able to qualitatively reproduce the same type of behavior as the human participants from Thompson et al. (2013) in our model, we left elements out of the current model that would have allowed fitting the data quantitatively (i.e. instructions reading). We could also refine the model and specifically the training procedure to simulate other versions of the tasks (drinking-age problem, negated version, etc.). Finally, the Wason task is limited; we would like to extend this framework to more complex real-life tasks where both automatic and reflective behaviors are required.

References

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

Evans, J. S. B. (2006). The heuristic-analytic theory of reasoning: Extension and evaluation. Psychonomic Bulletin & Review, 13(3), 378-395.

Johnson-Laird, P. N. (1980). Mental models in cognitive science. Cognitive science, 4(1), 71-115.

Juvina, I., Larue, O., & Hough, A. (2017). Modeling valuation and core affect in a cognitive architecture: The impact of valence and arousal on memory and decision-making. Cognitive Systems Research.

Larue, O., Poirier, P., & Nkambou, R. (2013). Hypothetical-thinking based on cognitive decoupling and thinking dispositions in a dual cognitive agent. Biologically Inspired Cognitive Architectures, 6, 67-75.

Leslie, A. M. (1987). Pretense and representation: The origins of "theory of mind.". Psychological rev, 94(4), 412.

Sun, R. (2015). Interpreting psychological notions: A dual-process computational theory. Journal of Applied Research in Memory and Cognition, 4(3), 191-196.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. Human Factors, 45(1), 61-76.

Taatgen, N. A., Rijn, H. van, & Anderson, J. (2007). An integrated theory of prospective time interval estimation. Psychological Review, 114(3), 577–598.

Thompson, V. A., Evans, J. S. B., & Campbell, J. I. (2013). Matching bias on the selection task: It's fast and feels good. *Thinking & Reasoning*, *19*(3-4), 431-452.

Wason, P. C. (1968). Reasoning about a rule. The Quarterly journal of experimental psychology, 20(3), 273-281.

ACT-R Workshop at MathPsych/ICCM 2018

Christian Lebiere[1] **(cl@cmu.edu), Dario D. Salvucci**[2] **(dds26@drexel.edu), Michael D. Byrne**[3] **(byrne@rice.edu), Niels A. Taatgen**[4] **(n.a.taatgen@rug.nl), J. Gregory Trafton**[5] **(greg.trafton@nrl.navy.mil)**

[1] Department of Psychology, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15208, USA
[2] College of Computing & Informatics, Drexel University, 3141 Chestnut St., Philadelphia, PA 19104, USA
[3] Department of Psychology, Rice University, 6100 Main St., Houston, TX 77005, USA
[4] Department of Artificial Intelligence, University of Groningen, Nijenborgh 9, 9747 AG Groningen, Netherlands
[5] U.S. Naval Research Laboratory, 4555 Overlook Ave. SW, Washington, DC 20375 USA

ACT-R (Anderson, 2007) is a cognitive architecture that provides a theory and simulation system for human cognitive, perceptual, and motor processes. As a cognitive theory, ACT-R aims to account for the detailed steps of thought and behavior as observed through standard behavioral data as well as brain-imaging data (see, e.g., Anderson et al., 2004). As a simulation system, ACT-R can be placed in the role of a virtual human user, performing a simulated (or even real robotic) task and allowing for direct comparison between human and model behavior. To date, ACT-R has been used to produce integrated models for hundreds of tasks ranging from basic experimental tasks to complex applied tasks; the ACT-R web site [http://act-r.psy.cmu.edu] lists the many models, publications, and researchers associated with the cognitive architecture.

This workshop serves to update both the ACT-R community and the cognitive modeling community at large about recent advances in the ACT-R architecture. For researchers already using ACT-R, the workshop provides a venue for presenting and hearing about recent changes and novel applications of the architecture. For others working with (non-ACT-R) computational cognitive models, the workshop provides an overview of the variety of application domains addressed by the architecture, and encourages sharing of ideas that would benefit ACT-R and other modeling frameworks alike.

Participation: Based on attendance numbers from similar past workshops held at the ICCM and Cognitive Science Conferences, we expect a registration of approximately 30 to 70 participants. Potential speakers include, in addition to the organizers, other regular members of the ACT-R community as well as more junior contributors. Reflecting a broadly disseminated call for participation, currently registered participants reflect a mix of experienced ACT-R modelers and interested modelers not currently part of the ACT-R community.

Format: The full-day workshop includes presentations on recent ACT-R developments from the user community, an update on current developments on the architecture itself, and panel discussions on current popular research topics and trends relevant to ACT-R. The schedule includes ample time for discussion and sharing of ideas among the attendees.

Program: The prospective schedule for the 2018 ACT-R Workshop consists of 3 sessions, each of which includes 3 20' talks (including questions) and a panel discussion, followed by a final session on recent and prospective developments to the architecture:

Learning and Transfer in Complex Environments

John Anderson	Transfer of Cognitive Skills
Frank Ritter	Using a Model to Predict Learning and Retention in a Large Study of a Complex Task
Christian Lebiere	Decision Making in the Presence of Deceptive Signals

Neural and Perceptual Embodiments

John Lindstedt	Simple Agglomerative Visual Grouping for ACT-R
Patrick Rice	Using TMS to Test the Associations between ACT-R Modules and Cortical Regions
Andrea Stocco	ACT-R Parameters from Resting State Neuroimaging Data

Human Machine Interaction

Greg Trafton	Two Models of Social Influence
Sterling Somers	CogXAI: Cognitively eXplainable Artificial Intelligence
Nele Russwinkel	Developing a Concept of an Active Self through Natural Interaction

Future of ACT-R

Dan Bothell	Software Updates
Everyone	Open-Ended Discussion

Publicity: We used the ACT-R mailing list and related lists (e.g., the Cognitive Science mailing list) to publicize the workshop as done in past years. We expect there to be additional attendees due to the publicity that comes as part of the MathPsych/ICCM meeting and hope to get attendees from laboratories that do not normally come to our standalone meetings. As we have for previous meetings, we will make all presentation content available on the ACT-R website.

Requirements: The workshop uses standard audio-visual projection for displaying computer talks and demos.

References

Anderson, J. R. (2007). How Can the Human Mind Occur in the Physical Universe? New York: Oxford University Press.

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

Comparing Models of Visual Search in Heterogeneous Search Fields

Stefan Lindner (stefan.lindner@campus.tu-berlin.de), Lennart Arlt (lennart.arlt@campus.tu-berlin.de), Nele Russwinkel (nele.russwinkel@tu-berlin.de)
Department of Cognitive Modeling in dynamic Human-Machine Systems, TU Berlin, Marchstr. 23
10587 Berlin, Germany

Abstract

The paper investigates visual search in heterogeneous search fields with the aim of capturing search times with cognitive models. An icon search experiment was conducted in which target-distractor similarity (low vs. high) and distractor-distractor similarity (low vs. high) of icons, target presence (present vs. absent) and the set size (6x4, 8x4 or 8x6 icons) were varied. At the same time a total of 6 ACT-R models - each implementing a different search strategy hypothesis - were created (4 cluster search models, 1 row model and 1 basic model) and their fit with the experimental reaction times assessed. All cluster models were able to fit the general pattern of reaction times fairly well, but varied in fit among different conditions. A cluster model assuming search along 2-by-2-icon clusters achieved the best overall fit.

Keywords: visual search; similarity; ACT-R; cognitive modeling.

Introduction

Whether we are looking for our car on a parking lot, our favorite socks in midst of our clothes or for a specific app on our phone - search in heterogeneous visual fields is a common task.

If the object we are looking for has salient features we are in luck, because our visual system allows for a parallel search in that case. This means that we can scan a large number of items and locations in our search field at once, making the target item "pop out" between other items. If that is not the case, however, we often have to switch to sequential search – looking at one item after the other. At the least we have to sequentially check some candidate items that stand out or share properties (Wolfe, Friedman-Hill, Stewart, & O'Connell, 1992). The most interesting case is heterogeneous search, in which only some items share characteristics with the target item.

Since visual search in such cases is a critical time component for applied research and evaluation of usability, it would be helpful to identify what strategies users follow to be able to predict search time in such cases. User models rarely focus on visual search, even though it would be beneficial to have a simple to apply modelling approach that is able to predict valid search times but is not too complex to apply.

Therefore, our goal is to capture visual search times in such tasks with models that make as few assumptions as possible. To this end, we conducted an experiment that had subjects look for a target item on a tablet screen. In a previous work we already presented an ACT-R model that attempted to fit reaction times in such a task (Lindner, Russwinkel, Arlt, Neufeld, & Schattenhofer, 2017). The original model assumed a search along rows and achieved fairly good fit. However, since this model is unlikely to scale properly with screen size, we created 4 new models that visually fixate clusters of items. These models aim to more closely depict fixations in the area of near-central foveal vision (Provis, Dubis, Maddess, & Carroll 2013). We compared the model predictions of 6 models in total: a "naïve" model, the row model and 4 cluster models. In the following we will first introduce the experimental setup and its results. Afterwards we will discuss the models and then compare their predictions with the experimental results.

Experiment

The paper presents new results from a recent experiment that was conducted as a follow-up to the study by Trapp & Wienrich (2018). The experiment thus uses a very similar setup. It was conducted to provide novel data that model predictions could be tested against.

In order to put the models to a strong test, the experiment mainly varies the heterogeneity of the search field. It presents item arrays that differ in the number of items that have the same color as the target item and in the number of distractor items that share the same color amongst themselves.

It also looks at different item set sizes. Larger set sizes (compared to the previous experiments) were specifically meant to test how well models scale with the increase in search field size. In particular, we chose to present the item sets on a horizontal screen orientation to test longer rows. The participants performed a visual search task on a 20x15 cm mobile touch device, in which they had to find a specific target icon within a set of distracting icons.

Each trial was performed in the following manner: After the target icon was shown for two seconds, a fixation cross was presented in the center of the screen to ensure a standardized gaze point for all participants. After the fixation cross disappeared, a set of icons was shown. When the target icon was present in the set, the participants had to find and select the target icon (by touch) as fast as possible. Whenever there was no target, they had to select a specific button at the lower screen to indicate the absence of the target icon. Subsequently, they received feedback on whether their answer was correct or incorrect. The reaction time was recorded for each trial and served as a performance measurement.

Figure 1: Experimental similarity conditions according to color (for demonstration; not original icons used)

The two main independent factors in the experiment were target-distractor similarity (TDS; low vs. high) and distractor-distractor similarity (low vs. high) (see Figure 1). The other main factor was the display size: 6x4, 4x8 and 6x8 item setups were tested.

Additionally, as in the original study, target presence was varied independently. Overall the experiment thus followed a 2x2x2x3 within subject design. Overall, 23 participants (12 female; mean(age)=29; sd(age)=4) were tested. Each participant completed 12 trials in each condition for an overall 288 trials.

The tablet was placed at a gaze distance of approximately 50 cm from the subject. The items were 1,5x1,5 cm in size and 0,5 cm apart from each other.

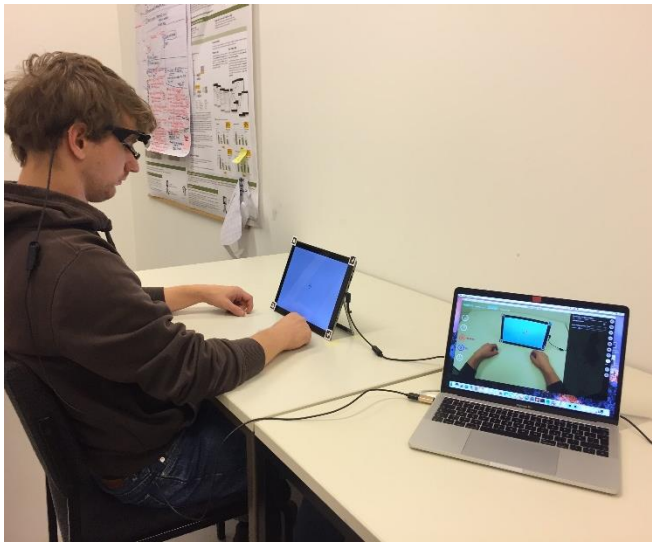


Figure 2: Experimental setup. The subjects interacted only with the tablet. Eye movement data was recorded via tracking glasses.

Experimental Results

We will discuss the experimental results only briefly, since the main focus of the paper lies in the modeling approach. Absolute reaction times are shown and discussed in the "Results and Model Fit" section.

Factor significance results largely replicated the findings from Trapp & Wienrich (2017). Graphs of the reaction times in all conditions is shown in the section "Results and Model Fit".

	F	dfs	p		ges
intercept	217.67	1, 22	<	.001	.69
TDS	47.62	1, 22	<	.001	.26
DDS	0.25	1, 22		n.s.	.00
target presence	12.88	1, 22	<	.01	.02
TDS:DDS	0.13	1, 22		n.s.	.00
TDS:target presence	5.54	1, 22	<	0.05	.02
DDS:target presence	3.74	1, 22	*	0.1	.01
TDS:DDS:target presence	0.01	1, 22		n.s.	.00

Figure 3: Experimental results (ANOVA). Dependent variable: Completion time.

Reaction times were significantly lower if target and distractors were of different colors (TDS high). They increased in the absence of the target item. We also found an interaction between target-distractor similarity and target presence: the absence of the target item lead to a larger increase in reaction times if at least some distractors matched the target color.

Modeling

Where & What System

ACT-R (Anderson et. al., 2004; Andserson, 2007) visual search makes use of two main buffers, the visual-location buffer and the visual buffer. These two buffers mirror the two subsystems of vision, the **where system** and **what system (**Byrne, 2001). The where system simulates preattentive processes and relies on well accepted theoretical concepts (Wolfe, 1994; Treisman & Gelade, 1980). Each visual item has features such as type (text, or oval for a button or others), color or width. On a preattentive process stage it is possible to search for visual locations that contain items with a specific feature. After the visual location is identified attention can be located in this position to identify the item and all its features. The first process needs no time, the second process does need time. A shift of visual attention takes 135ms - 50ms for the production to fire that elicits the request of the shift and 85ms for the shift itself.

Modeling Principles

To keep models simple and to avoid overfitting, each model assumes, in principle, the same strategy for all experimental conditions and screen sizes.

All models also assume that items that share the same color as the target item can be globally located (e.g. if the target is yellow the model can find yellow items on the entire screen without previously encoding any part of the screen). They then make different assumptions about the size of the area in which all visual features of the candidate items within can be extracted with one fixation.

Single Model

This model separately encodes every single visual object that has the target color. It thus implements the simplest plausible mechanism and serves as a baseline.

Row Model

This model was already reported in Lindner et al. (2017). It uses a hybrid mechanism of visual search. It completely encodes each row that contains at least one item of the target color with one fixation. Despite achieving a good fit on display sizes with rows of length 4, it is unlikely to scale well with increasing screen size length since its predicted times are invariant to row length.

Cluster Models

These models are the main novelty of the paper. They attempt to improve in the row model by assuming the ability to search rectangular item clusters with a single fixation. This is psychologically more plausible as it mirrors foveal vision: It also is better fit to recreate growing reaction times with growing screen sizes. The models assume a cluster size of 2x2 (4-item cluster models) and 1x2 (2-item cluster models).

Specifically (and very similarly to the row model), these models fixate a *candidate item* (an item that has the same color as the target item) at random and scans all items within the cluster that contains this item with the same fixation. If the target item is not contained in the cluster, the next candidate item and its cluster are fixated. Once a cluster has been fixated, the model does not return to that cluster. If all clusters that contain candidate items have been searched, the model decides that the target is not present.

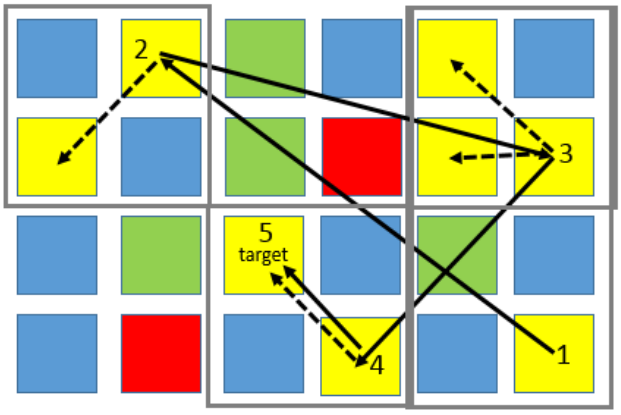


Figure 4: Exemplary search path of the 2x2-cluster model. Digits denote fixations, solid arrows denote saccades, dotted arrows denote peripheral scanning of additional candidate items in a cluster. Only one item is fixated per cluster (except for the cluster that contains the target item - if it is not the item that is targeted first in that cluster)

Extended Cluster Models

The extended cluster models work slightly differently from the cluster models. They assume a steadier scan path from the upper left line-wise to the bottom. They also "scan" every cluster, requiring the time of one ACT-R production if no candidate item is contained in a cluster. If a candidate item is contained in a cluster, the models work in the same way as the cluster models: they require one fixation to scan the entire cluster. The ideas behind the extended cluster models was to account for possible non-attentional scanning of noncritical screen areas. It is also an approach that somewhat scales with set size independently of the number of candidate items.

Model Specifications

It should be noted that the 4-item cluster model and the extended 4-item cluster model (as well as the single and the row model) were created and used to predict reaction times prior to the experiment. Since the previous row model had suggested that an encoding of 4 items per fixation seemed reasonable, we expected the 4-item cluster models to make the best predictions.

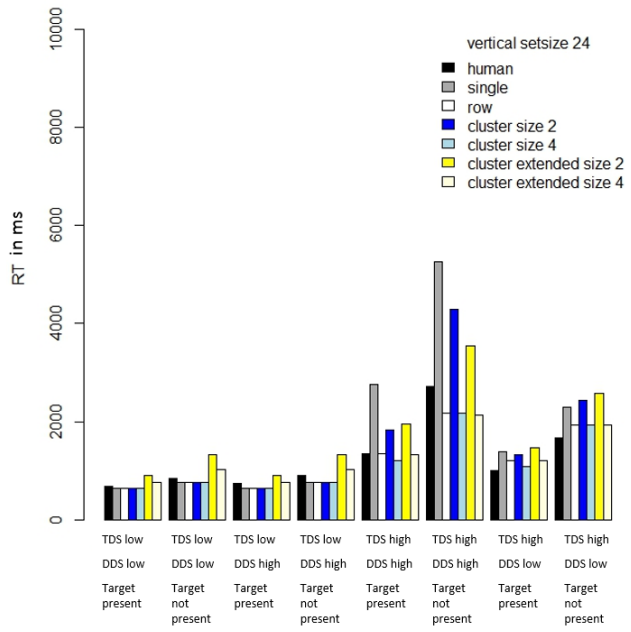
The two 2-item cluster models were created after the experiment to test out the parameter space somewhat. The models were not otherwise fit.

In the models no ACT-R parameter values were changed. The declarative memory consists of a goal chunk and a chunk that stores color, text and width of the current target item. All presented models and the GUI are published online at https://depositonce.tu-berlin.de/handle/11303/7441. To obtain the simulation results, each model was run 1000 times in each condition.

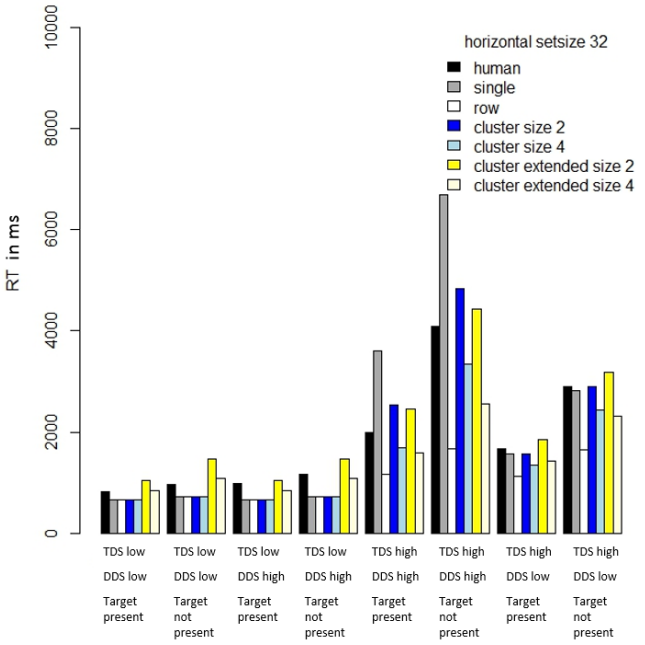
All models are written and commented in a way that should make it easy to adapt them to new situations. In the cluster models the size of the clusters can be given in terms of *nxm* items and the absolute size of the cluster will be adjusted automatically.

Results and Model Fit

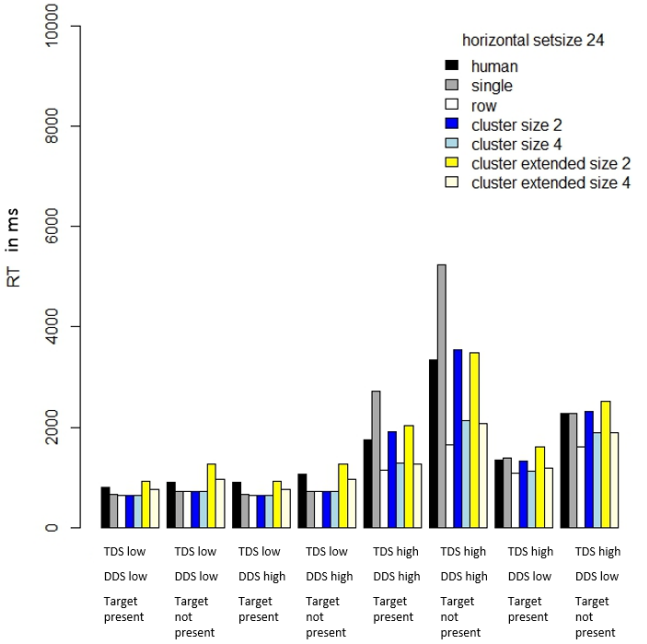
In the following graphs, all reaction times and model predictions are presented for each set size separately. In order to put the models to the strongest possible test, we also report the reaction times of the 4x6 screen size from the Trapp & Wienrich (2018) experiment. Underneath each graph we also report the correlation of each model with experimental data as a measure of relative fit and RMSE and RMSSD absolute fit measures (Schunn & Wallach, 2005).



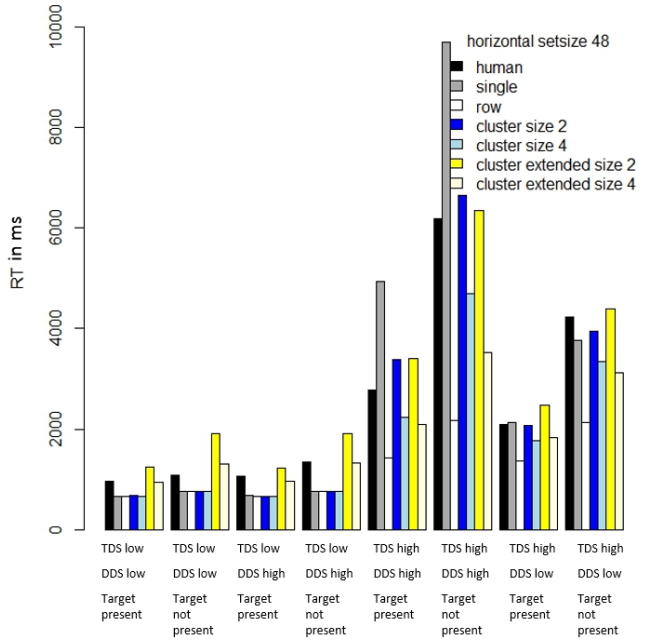
Setsize v24	single	row	cluster size 2	cluster size 4	cluster extended size 2	cluster extended size 4
RMSSD	4.69	1.50	3.04	1.39	4.18	1.43
RMSE	707 ms	155 ms	436 ms	153 ms	374 ms	167 ms
correlation	0.977	0.938	0.996	0.946	0.987	0.939



Setsize h32	single	row	cluster size 2	cluster size 4	cluster extended size 2	cluster extended size 4
RMSSD	5.31	5.37	3.86	3.94	4.35	2.56
RMSE	652 ms	619 ms	232 ms	341 ms	192 ms	359 ms
correlation	0.946	0.951	0.986	0.996	0.992	0.977



Setsize h24	single	row	cluster size 2	cluster size 4	cluster extended size 2	cluster extended size 4
RMSSD	4.66	4.81	3.80	4.32	3.26	2.60
RMSE	456 ms	414 ms	119 ms	304 ms	131 ms	297 ms
correlation	0.969	0.941	0.995	0.970	0.993	0.963



Setsize h48	single	row	cluster size 2	cluster size 4	cluster extended size 2	cluster extended size 4
RMSSD	4.99	5.39	3.63	3.91	3.55	2.57
RMSE	881 ms	1012 ms	241 ms	424 ms	273 ms	621 ms
correlation	0.949	0.950	0.988	0.996	0.992	0.978

Relative Fit (Correlations):

All models show a high relative fit. This is rather unsurprising since the general workings of all models are rather constrained. In all models the number of fixations grows proportionally to the number of candidate items. They therefore mainly differ by how large that proportional growth is. It is therefore of more interest to look at absolute times.

Absolute Fit (RMSSD and RMSE:

The "naïve" single model shows a large overall deviation in all conditions, demonstrating the need for more sophisticated models. As was already established in Lindner et al. (2017) the row model provides a very good fit for the vertical 24-item set size. However, as expected it fails to properly scale with larger row sizes, both in the horizontal 24-item screen and the 32- and 48-item screens.

The 4-item cluster model, while scaling better with size still falls short in two main ways. It scales a little too slowly with set size, falling consistently below experimental reaction times, especially in set size 48. This suggests that maybe on average a little fewer than 4 items can be encoded at once. It also fails to capture the slightly increasing reaction times in condition 1 and 2 and the slightly longer reaction times when the target is absent in these conditions.

The 2-item model seems to scale somewhat better with size, but has the same shortcomings as the 4-item model in conditions 1 and 2. All models discussed so far make the assumption of an immediate pop out independent of set size in conditions 1 and 2, an assumption that is not fully backed up by the data.

The extended cluster models do better in this regard, as they assume a growing number of decisions (productions) with growing set size – independently of the number of candidate items. The extended 2-item cluster model shows this scaling, but unfortunately now overpredicts reaction times in condition 1 and 2. Just like the 2-item cluster model it shows a very good scaling with growing set size and a high number of candidate items (conditions 3&4). This suggests that participants where roughly able to scan 2 items with one fixation once they looked at a spot with at least one candidate item.

Judging by RMSSD, however, the extended 4-cluster model performs best overall. It matches the other 4-item models in the vertical 24-item condition but fits the data significantly better than all other models in the other conditions. It does especially well in conditions 1 and 2 across all conditions. This suggests that participants still somewhat scan the screen without "committing" visual fixations even when the target item stands out. On the large screens, however, the model underpredicts the growth in reaction times.

Discussion

Inherent to our modeling approaches was the idea that search behavior in all conditions is created by the same mechanisms. In reality, that might not be the case, however. For example, with growing screen size, participants might lose track of the already covered search areas more often. This in turn might lead them to double check more often or to scan more carefully. This could lead to different search strategies that depend on screen size and frequency of candidate items. Judging from the model fits, it seems that participants scan about 4 items with one fixations when few target items are present. On larger screens the models suggest the capture of about 2 items per fixation. Adaptively choosing models in this way should provide an adequate fit for practical purposes, especially in models that have a visual search component that is not the main focus of the task. The next steps could be the creation and testing of such an adaptive model and the analysis of the eye tracking data in regards to fixation paths.

Acknowledgments

We would like to thank Lisa-Madeleine Dörr for creating multiple ACT-R GUIs for our models.

References

Anderson, J. R. (2007). *How Can Human Mind Occur in the Physical Universe?* New York: Oxford University Press.

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

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.

Lindner, S., Russwinkel, N., Arlt, L., Neufeld, M. and Schattenhofer, L. (2017). Modeling of Visual Search and Influence of Item Similarity. In M. van Vugt, A. Banks & W. Kennedy (Eds.), *Proceedings of the 15th International Conference on Cognitive Modeling*. Coventry, United Kingdom.

Provis JM, Dubis AM, Maddess T, Carroll J. Adaptation of the Central Retina for High Acuity Vision: Cones, the Fovea and the Avascular Zone. *Progress in retinal and eye research*. 2013;35:63-81. doi:10.1016/j.preteyeres.2013.01.005.

Schunn, C. D., & Wallach, D. (2005). Evaluating goodness-of-fit in comparison of models to data. *Psychologie der Kognition: Reden und Vorträge anlässlich der Emeritierung von Werner Tack*, 115-154.

Trapp, A. K., & Wienrich, C. (2018). App icon similarity and its impact on visual search efficiency on mobile touch devices. *Manuscript submitted for publication.*

Treisman, A. M., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology, 12,* 97–136.

Wolfe, J. M., Friedman-Hill, S. R., Stewart, M. I., & O'Connell, K. M. (1992). The role of categorization in visual search for orientation. *Journal of Experimental Psychology: Human Perception and Performance, 18(1)*, 34-49. http://dx.doi.org/10.1037/0096-1523.18.1.34

Wolfe, J. M. (1994). Guided Search 2.0: A revised model of visual search. *Psychonomic Bulletin & Review, 1(2)*, 202–238.

Simple agglomerative visual grouping for ACT-R

John K. Lindstedt (j.k.l@rice.edu)
Department of Psychology, 6100 Main Street
Houston, TX 77005 USA

Michael D. Byrne (byrne@rice.edu)
Departments of Psychology and Computer Science, 6100 Main Street Houston, TX 77005 USA

Abstract

The human visual system tends to group related objects in the environment, allowing for more efficient use of attention, but sometimes leading to critical errors in executing a task. ACT-R's vision module currently has no concept of visual grouping, per se. We present both theoretical and practical motivations for imbuing ACT-R with visual grouping processes, and then walk through our implementation of a simple, minimally disruptive, generally applicable, and extensible system for assigning visual objects groups based on proximity, accounting for both spatial and temporal extension. Code is available and implications are discussed for employing the visual grouping system in ACT-R models. Finally, we discuss the system's limitations, extensibility, and its future development.

Keywords: cognitive modeling; visual grouping; UI; voting; human factors; model generalizability

Introduction

The human visual system employs visual grouping to more efficiently interact with the environment. Related elements are considered together, enabling our cognitive systems to shift attention to or away from groups of visual elements that are related in some meaningful way. But, as with any human system, there are kinks in how the visual grouping system works that cause us to make occasional, sometimes critical, errors in how we parse the world around us.

An example of how visual groups appear to impact task performance lies in the literature on voting ballots, wherein some research has been done demonstrating humans' sensitivity to the layout of the screen information over time. A Brennan Center for Justice report titled "Better Ballots" (Norden, Kimball, Quesenbery, & Chen, 2008) highlights a variety of voting ballot designs that produced voting errors, such as omitting a vote or voting twice. Figure 1 shows one such "bad" ballot, wherein so many voters in one precinct skipped a specific race on the second screen that it changed the outcome of the congressional election. Initially, one might suspect this error could be due to distraction by extreme salience, such as the bold, colored header drawing the attention down to the second race on the page. But according to Greene (2010), the effect appeared to be due the number or arrangement of races (and instructional elements) on the previous screen:

> While the highlighting of race headers did not reliably predict initial omissions of the critical race, the number of races presented on the first voting screen did: when voters saw two races on the first screen, they were less likely to omit the critical race on the following screen than were voters who saw only a single race on the first screen.

Indeed, it appears that the culprit in this particular ballot was the difference in layouts between the two screens, in which visual grouping processes presumably play a large role.

As the ACT-R cognitive architecture (Anderson, 2007) is often used in human factors applications for the purposes of evaluating interfaces and predicting performance, we believe this is precisely the sort of error ACT-R should have the capacity to predict. However, at present the ACT-R's "visicon" system, for all of its features, lacks any concept of visual grouping– it simply lists all of the available visual objects in isolation. Our intention with the present work is to implement a simple, consistent, and transparent method of grouping visual elements in ACT-R that works for any task and does so in a minimally invasive manner.

Motivations

Implementing a visual grouping algorithm for ACT-R has both theoretical and practical value. On the side of theory, these are well-documented processes that occur in human vision and if implemented correctly, can improve the validity and plausibility of models written in ACT-R. In terms of practical value, making visual groups available to modelers would offer more generalizable models, as well as a handful of conveniences for interacting with the visicon.

There is already much work on the human processes involved in segmenting a visual scene into separate visual groups. Rosenholtz et al. (2009) present a model that synthesizes a variety of visual features to simulate human visual grouping. Those features include proximity, similarity of color or luminance, continuity, and orientation, among others (and offers a nice review of relevant research on each). Their model:

> "... translate[s] a complicated two-dimensional image, in which segmentation is difficult, into a higher-dimensional representation where straightforward methods yield good results. Our particular technique uses a high-dimensional blur operation, which is simple to implement and understand."

Our work borrows– if not their specific techniques– their philosophy by taking a handful of the features available in the visicon and using them to perform a simple and clear process to segment them into visual groups.

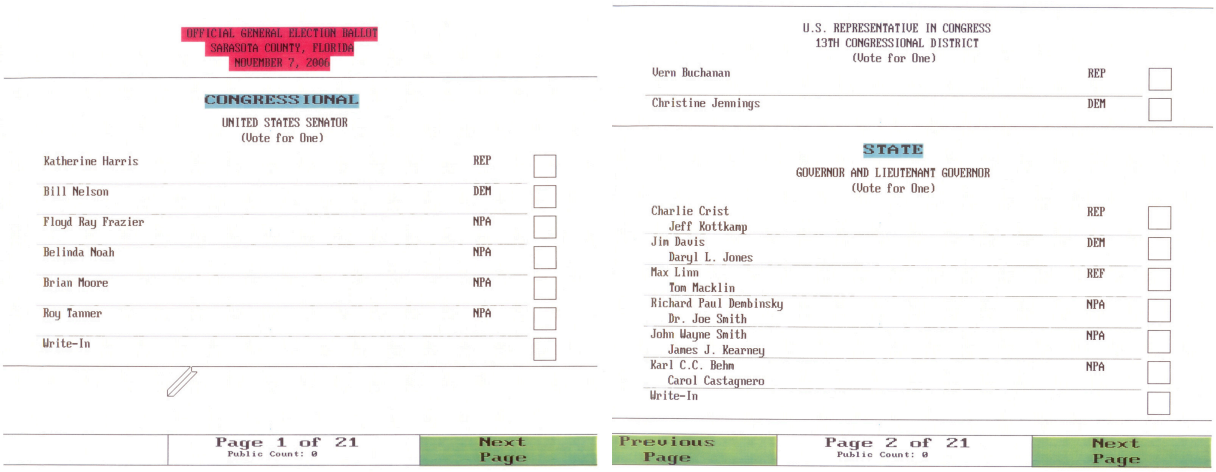


Figure 1: Two screen captures from the 2006 Sarasota County electronic voting system (first screen on the left, second screen on the right). So many voters failed to notice the race for U.S. Representative (top of right) that it changed the result of the election for that race. This error is thought to be due to the layout not accounting for human error due to visual grouping.

The introduction of a visual grouping system also fixes some problems and enhances some functionality in ACT-R modeling. Models often need to search the visual environment for particular elements in order to proceed with their tasks. Current modeling solutions for visual search tend to involve either (a) searching for the nearest unattended visual object, or (b) using specific SCREEN-X and SCREEN-Y coordinates to restrict that search. These solutions tend to suffice, but are problematic. In the first case, depending on the scan-path already taken, the model may find a visual location that is nowhere near the relevant section of the screen simply because it has already examined other nearby elements. In the second case, we are hard-coding knowledge of the screen layout, somewhat decreasing the model's cognitive plausibility and preventing it from generalizing across screen layouts without continuously spoon-feeding it special task knowledge. By introducing visual grouping, modelers can both restrict visual search to a particular region of the screen, and also build their models in such a way that the model should be able to locate the task-relevant regions of the screen regardless of their particular configuration and layout.

Visual grouping algorithm

In developing the visual grouping algorithm, we wanted a method that would require the fewest parameters from the modeler. An obvious, and potentially parameter-free, option would be the well-established *k*-means clustering method. Im et al. (2016) find success in developing a model of visual grouping using *k*-means. However, early in development we decided to go a different direction because: first, *k*-means and other density-based clustering methods are best suited for displays featuring masses of simple points, whereas most tasks modeled in ACT-R tend to involve more sparse, highly structured visual objects that possess width and height. Second, these clustering algorithms also tend to involve an element of randomness (e.g., bootstrapping) which can be quite frustrating for the purposes of writing and even understanding our own models. Though it is possible there are some instabilities in the way humans actually group visual objects, it is unclear whether the kind of uncertainty produced by these clustering algorithms would bear any resemblance to the uncertainty in the human visual system.

Instead, we decided to start with a method that is more consistent (for our modelers' sake) and transparent (for our developers' sake). In the same vein as Rozenholtz et al. (2009), we seek to apply relatively simple, general methods to the task of segmenting a visual scene into meaningful groups, provided with the representation of the visual objects already present in the ACT-R visicon. In the first version of our system, we account only for a visual object's positional features, SCREEN-X and SCREEN-Y, and its features of spatial extent, WIDTH and HEIGHT. We then further propose a system for how these visual groups propagate through time.

Interested readers can access our visual grouping code at the following github repository:

https://github.com/john-k-lindstedt/visual-grouping-actr

Installation is as simple as dropping the visual-grouping.lisp file into the user-loads folder within the ACT-R file tree. From there, the modeler needs only to specify the grouping radius and collision type desired, and the groups will be automatically generated and seamlessly added to objects in the visicon.

Integration with ACT-R

Our implementation of the visual grouping algorithm functions by intercepting the list of visual features for all of the elements of the scene used by ACT-R before the visicon is constructed, determining the visual groups for those elements,

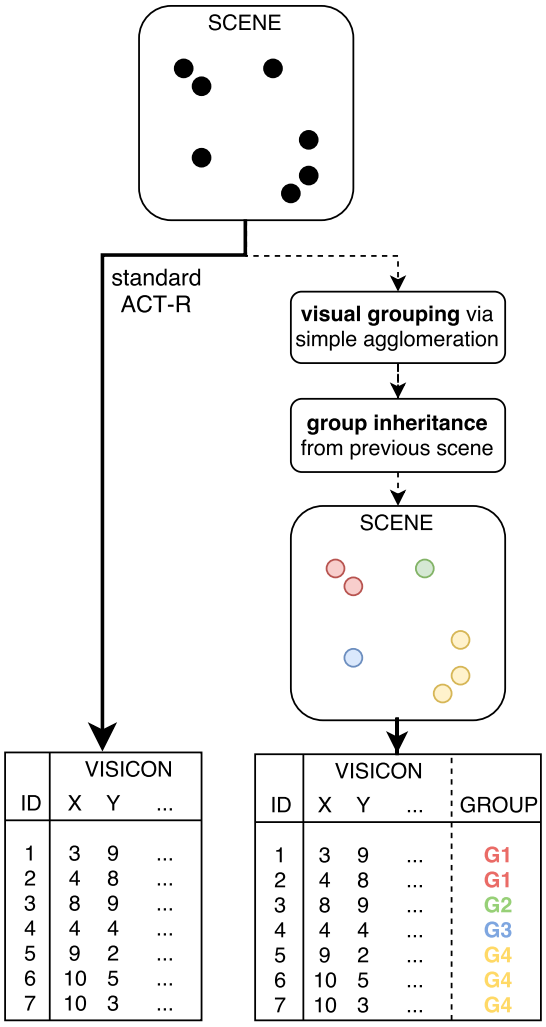


Figure 2: Flow of information from the visual scene to the visicon by default (left), and through our visual grouping system (right). The process is purely additive, giving each visicon entry a GROUP slot, and not disturbing any of the others.

and then returning that list intact but with the new group information attached. ACT-R then constructs the visicon as normal, but each visual location now has a GROUP slot that can be used in visual location requests like any other. Figure 2 illustrates this process.

Visual grouping by simple agglomeration

To determine which visual objects belong to which visual groups, we use a method called "simple agglomeration." Each group begins with a single visual element and recursively adds other nearby elements by checking for "collisions"; i.e., whether any nearby elements are within a "grouping radius" (a parameter presently left to the modeler to adjust). Two methods of collision detection are available: point-collision and box-collision. Ultimately, each object on the screen is assigned a unique and arbitrary symbol corresponding to its visual group.

Point collision method: Simple and fast grouping The point-collision method is minimalist: simply check whether the screen coordinates of two visual objects is within the grouping radius:

```
point-collision(obj1, obj2, radius):
1. if distance(obj1,obj2) < radius,
   obj1 and obj2 have collided,
   return true
```

This method is simple to calculate, but does not account for an object's size on the screen. As such, it is best used when objects are of similar sizes and shapes (i.e., singular characters or symbols), or when the screen is very dynamic and needs to update often. Figure 3 (top left) depicts this collision method.

Box-collision method: Accounting for extension in space Many objects in user interface displays have meaningful extension in space (text, buttons, images, etc), so we also implemented a box-collision method that accounts for an object's width and height by determining if the nearest point on one object's bounding box is within the grouping radius of the edges of another object's bounding box:

```
box-collision(obj1, obj2, radius):
1. target = the nearest location to obj1
   on the perimeter of obj2's bounding box
2. check point-collision(corner, target) for
   each corner of obj1's bounding box
3. check if target is within radius of the
   top, bottom, left, or right of obj1's
   bounding box
4. check whether target is overlapping with
   obj1's bounding box (just in case!)
5. if any of the above is true, obj2 and
   obj2 have collided, return true
```

The box-collision method requires more computation, but accounts for width and height in a more realistic way than the point-collision method. In practice, both the point-collision and box-collision methods are quite fast, and many tasks modeled in ACT-R (especially user interface tasks) involve static displays with relatively few elements. As such, we recommend the box-collision method over the point-collision for most applications. Figure 3 (top right) depicts this collision method.

Growing visual groups via simple agglomeration The simple agglomeration method begins with a visual group containing a single visual object and then "grows" that group by iteratively adding nearby points until none remain within the grouping radius. Then a new, unexamined visual object is selected, and the process is repeated until every object is assigned a group:

```
grouping(scene):
1. find an unexamined point, a, in the scene
2. find another unexamined point, b, in the
   scene
```

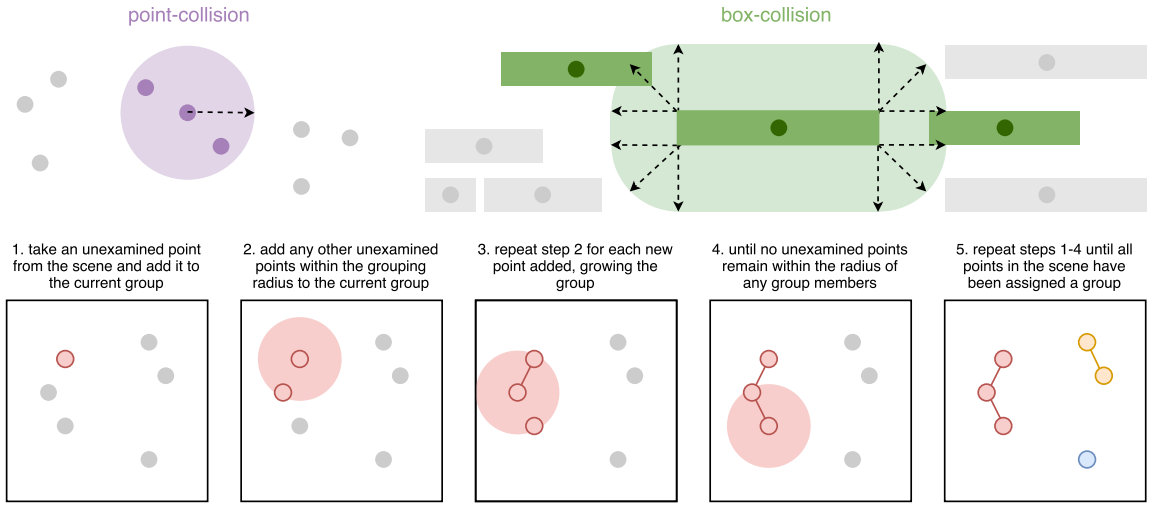


Figure 3: The two methods of collision detection available. Point-collision (top-left) is relatively fast and simple in that it involves a single comparison, examining whether the distance between two points is within the grouping radius. Box-collision (top-right) is more complex, as it involves several more comparisons and detecting the nearest point on a neighboring visual object's bounding box, but it allows for all objects' spatial extent to be considered when determining visual groups. The bottom half of the figure illustrates the steps of the simple agglomeration grouping method.

```
3. if collide(a,b,r): add b to the group
4. grow the group by repeating steps 2 and 3
   for each new point added until there are
   no more nearby points
5. assign all members of the group a
   new group-id
6. repeat steps 1-5 until all points in the
   scene have been examined, and all points
   now have an associated group-id
```

Figure 3 (bottom) depicts this process visually. Figure 4 shows a sample output of the visual grouping algorithm as applied to a screen of our VoteBox system. It is notable that the order in which points are added to groups is irrelevant– all point collisions are mutual and group growth is both strictly additive and exhaustive, so there is no "competition" to speak of between groups.

Inheritance: visual grouping extended in time

We also want to allow visual groups to extend in time, lest the model be forced to study a new set of visual groups every time the display is processed. To achieve this, we employ the same collision-detection method used in the simple agglomeration method to detect whether visual groups in a new scene overlap with any known groups from the previous scene. We achieve this with the following steps:

```
inheritance(current-scene, previous-scene):
1. count the number of unique group-wise
   overlaps between each pair of groups in
   current-scene and previous-scene
2. a current group inherits a previous
   group's group-id only if both the current
   group and a previous group exclusively
   overlap with one another
3. assign a new group-id if:
3a. the current group is new, i.e. it
    does not overlap with any previous groups
3b. the current group is the result
    of a merge, i.e. it overlaps with more than
    1 previous group
3c. the current group is the result
    of a split, i.e. it overlaps with a previous
    group that also overlaps with at least one
    other current group
4. if a previous group overlaps with no current
   groups, that group is dead and its group-id
   will not propagate forward in time.
```

Note that, at present, we assume that the only way a group can directly inherit a previous group's identity is for both the previous and current group to have mutually exclusive overlap. All other cases are considered to be "confusing," and generate new group IDs (likely triggering the model to need to re-study the screen layout before proceeding). The temporal duration and propagation of these visual groups is an open research question, which we discuss in the next section.

Figure 5 demonstrates how the visual groups would propagate in an example bearing some resemblance to the Sarasota ballots mentioned above. The ability for visual group identities to be inherited over subsequent scenes enables models to make the same kinds of errors as voters did in the Sarasota congressional election– when a task-critical section of the screen (i.e., a congressional race) directly inherits its group identity from a segment containing only task-adjacent information (headers or instructions), the model can mistakenly

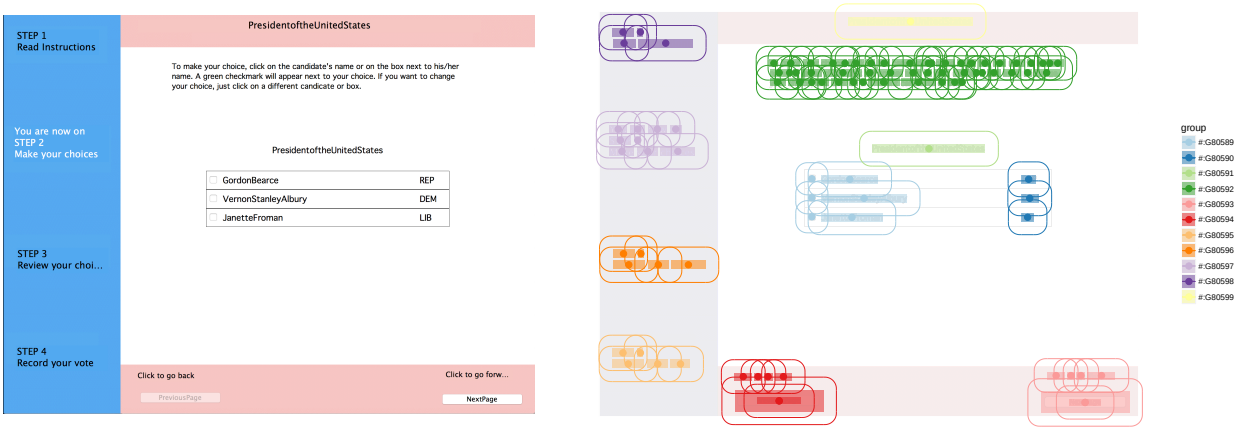


Figure 4: An example outcome of the visual grouping algorithm. The VoteBox task environment (left) is a relatively simple display with visual elements distributed somewhat sparsely across the screen. The right depicts the VoteBox task environment (dimmed) with the visual groups (colors) produced by the simple agglomeration method (grouping radius = 25 pixels) layered over top. For each visual location, the points represent the position, the shaded boxes represent the width and height, and the curved lines indicate the radius within which a box-collision occurs.

consider the task-critical segment as irrelevant and skip it entirely.

Implications for modeling

One of our goals in developing a visual grouping algorithm for the visicon was to provide functionality with as little disruption as possible to the way the ACT-R visicon works. As such, our implementation is designed to work exclusively with the information the visicon would already use, requiring no custom-built devices whatsoever. We also wanted to be sure the system was, by design, minimally disruptive to existing models. Because the algorithm does not remove or alter any information from the default visicon, modelers will not need to rewrite any of their models in order to install and start using the visual-grouping system.

Concerns for ease of use aside, modelers still must introduce new routines to their models to make use of the visual groups provided. It is important to note that the group IDs generated by the visual grouping system are intentionally generic and anonymous, meaning they strictly cannot impart any task-relevant information. Instead, models will need to use knowledge of the task at hand (such as labels of relevant buttons and information about the stimuli expected) to study the scene and commit to memory (i.e., the imaginal buffer) the relevant visual groups in the scene. Models will also need to return to this state to re-study the scene should the interface undergo any radical changes between parts of the task.

In exchange for the effort of having to study the screen, models will no longer need to be imparted with special knowledge about relevant screen locations, thereby becoming inherently more robust to changes in the layout of the screen–so long as the same buttons are used and the same kinds of information are presented, a model that studies the screen first will always know how to do its task.

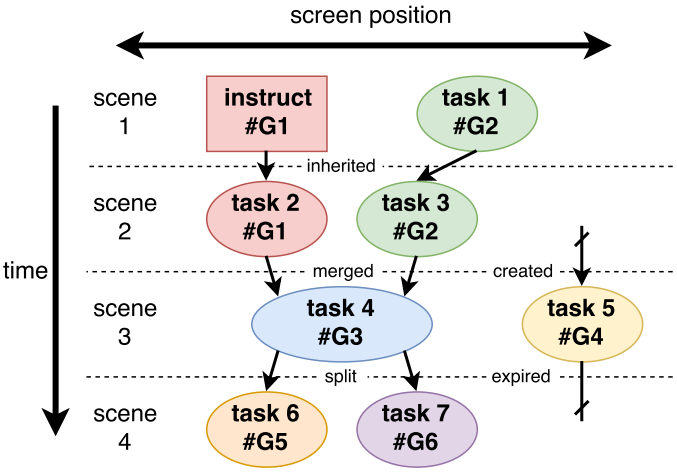


Figure 5: Sample scenario demonstrating how visual grouping would play out over multiple scenes in sequence. The first scene consists of an instructional portion and a task portion, assigned groups #G1 and #G2 respectively. In the second scene, task 2 inherits #G1 (the same group as the instructions, likely causing the model to skip task 2 entirely), and task 3 inherits #G2, despite being slightly offset from task 1. In scene 3, task 4 is considered the result of a merge between tasks 2 and 3, and is assigned the new group #G3, while task 5 is in a position that does not overlap with anything from the previous scene, so it receives the new group #G4. Finally, in scene 4, both tasks 6 and 7 are considered the result of a split from task 4 and receive the new groups #G5 and #G6, while group #G4 expires due to overlapping with no other groups.

Future directions

This visual grouping algorithm is meant to be a "first draft," initially aimed at addressing a handful of specific human error phenomena in the relatively simple domain of voting system usability. There is much work yet to be done, and many directions available for further development.

Currently, the appropriate grouping radius is left up to the modeler to determine, likely through trial and error (clearly not ideal). To provide a better default value for this radius– or an equation for finding one– questions such as "are groupings sensitive to scale?" and "what is the role of the retina?" will need answers either from a deeper delve into the visual grouping literature or by performing a battery of simple empirical studies.

Similarly, many questions remain about how exactly group inheritance functions. Currently, we assume that phenomena like splitting or merging two groups results in a sort of "confusion" for the model, so a new group label is applied. Instead it may be the case that there ought to be a "winner" in such an event: perhaps the largest group inherits? Or the group with the most overlap? Other questions about inheritance include: do group identities have any sort of "memory," recovering after a period of not being present? Are groups always constructed the same way, or is there an element of uncertainty we need to capture? Under what circumstances do we perceive visual groups as having "moved," rather than as two distinct groups? Do these processes function the same way when the screen is updating in real time as opposed to static, self-paced scenes? Simple empirical studies can illuminate how visual groups propagate forward in time, and eye tracking studies will help to illuminate what triggers humans to begin re-studying the screen.

We would also like to expand the scope of the visual features available for determining groupings beyond simple position and spatial extent. We believe our efforts are compatible with Rosenholtz et al.'s (2009) model, and as such we can tap into the existing literature on visual grouping, and their model in particular, to achieve even more robust visual groupings using a wider array of visual features, such as: contrast, color, luminance, orientation, continuity, etc. Representing these features, as well as SCREEN-X and SCREEN-Y, as a vector of numerical values would allow us to use a nearly identical measure of euclidean distance to detect "collisions" and classify groups in more interesting ways than simple spatial proximity.

We will also need to investigate to what extent humans employ hierarchical grouping. Dividing lines and containing boxes are commonly used by interface designers to partition the screen (like those shown separating the sections in Figure 1), but there are clearly smaller sub-groupings of information within those regions. Currently, our system can achieve something to the effect of identifying super- and sub-groupings by using more than one visual grouping radius, though the method for determining those radii will require thorough exploration.

Conclusion

Visual grouping processes help humans make sense of their visual environment. ACT-R, by default, lacks any sense of visual grouping. We attempted to remedy this because both (a) visual grouping is a well-documented phenomenon that explains certain elements of human behavior, and (b) the use of visual grouping offers modelers some practical conveniences and improvements to the generalizability of their models.

The visual grouping system we have implemented is a first pass at the problem, but still succeeds in many ways: the system is straightforward (we believe), it is minimally disruptive to existing models, it is stable in that it always produces the same visual groupings given a particular display, and it is extensible. In particular, we see the ability for models employing visual grouping to generalize across different screen configurations as a contribution to the overall plausibility of cognitive models in ACT-R.

Future empirical work and reviews of the literature will address: the extent to which the model can reproduce human errors on voting ballots, investigating the mathematical nature of visual grouping, and expanding the capabilities of the system to fit the needs of the modeling community. To that end, we extend an open invitation to interested modelers to propose– or implement– any desired additional features or alternative methods as we continue to develop this visual grouping system.

Acknowledgments

This research was supported by grant #CNS-12550936 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.

Greene, K. K. (2010). *Effects of multiple races and header highlighting on undervotes in the 2006 sarasota general election: A usability study and cognitive modeling assessment*. Doctoral dissertation, Department of Psychology, Rice University, Houston.

Im, H. Y., hua Zhong, S., & Halberda, J. (2016). Grouping by proximity and the visual impression of approximate number in random dot arrays. *Vision Research*, *126*, 291–307.

Norden, L., Kimball, D. C., Quesenbery, W., & Chen, M. C. (2008). *Better ballots.* New York, NY.

Rosenholtz, R., Twarog, N. R., Schinkel-Bielefeld, N., & Wattenberg, M. (2009). An intuitive model of perceptual grouping for HCI design. In *Proceedings of the 27th international conference on human factors in computing systems - CHI 09* (p. 1331). New York: ACM.

Modeling Perceptual Judgement in Believable Agents: A Signal Detection Approach

Spencer K. Lynn (slynn@cra.com)
Human Effectiveness Division, Charles River Analytics, Inc.
625 Mt. Auburn St., Cambridge, MA 02138

Taylor Curley (taylor.curley@gatech.edu)
School of Psychology, Georgia Institute of Technology
J.S. Coon Bldg., 654 Cherry Street, Atlanta, Georgia 30332

Peter Weyhrauch (pweyhrauch@cra.com)
Human Effectiveness Division, Charles River Analytics, Inc.
625 Mt. Auburn St., Cambridge, MA 02138

Abstract

Computer modeling of Warfighter performance is an increasingly important element for the Department of Defense in developing and evaluating tactics, techniques, and procedures (TTPs) as well as military acquisition strategies. To do this modeling, human performance researchers are working to integrate models of complex cognitive and physical systems, as well as the processes that moderate them, in a unified framework. To better model visual detection and identification processes in realistic performance situations, we have constructed an agent-based model of visual perception based on signal detection theory that can be moderated by exigent processes, such as stress and fatigue.

Keywords: Perceptual Judgement, Signal Detection, Believable Agents, Solder Simulation.

Introduction

A Warfighter in a combat environment is expected to continuously search his or her visual field to maintain situational awareness. Misidentification of relevant stimuli, such as failure to detect an enemy combatant or incorrect identification of a friend as an enemy, has costly results for the Warfighter and associated team members. Furthermore, Warfighters are often trained to sustain their attention over long periods of time, but by-products of situational demands, such as operational stress and fatigue, can significantly impact performance related to visual attention processes (Staal, 2004; Janelle & Hatfield, 2008). Thus, it is of paramount importance to understand perceptual judgment processes in individual Warfighters when confronted with moderators of operational performance.

Methods

The DREEMS Project

To better understand the dynamics of Warfighter performance, Charles River Analytics has introduced the Dynamic Representation for Evaluating the Effect of Moderators and Stress on Performance (DREEMS) project. Using a modeling language called Hap (Loyall et al., 1991), DREEMS models individual Warfighter performance through the use of situational awareness modeling (SAMPLE; Zacharias et al., 1996). The architecture models agent behavior as the cumulative result of an information processing module feeding into a situation assessor, which then guides an agent's decision-making via behavior trees. Moderating variables can exert influence over these modules at any point in the architecture.

Signal Detection Approach

Using a signal detection theory (SDT) approach to cognition provides a robust method of exploring a wide variety of behaviors, particularly for those in which individuals encounter perceptual uncertainty and behavioral risk (Lynn & Barrett, 2014). Importantly, the parameters of SDT mechanisms have been shown to be sensitive to moderating variables, such as emotion on field of view (Schmitz et al., 2009). Here, we follow previous research illustrating how perceptual decision parameters in a perceptual SDT model are moderated by differences in individual state (e.g., Lynn et al., 2012) and extend this approach to modeling visual threat detection in a battlefield environment.

Model Specifications

We model perception as a set of underlying receptors that correspond to different regions in the Field of Regard (FOR), or visual field, a span of 135°. These receptors respond to signals from the visual environment. When a signal is presented in the visual field, the ability of the agent to discern the signal from background noise is a function of the location of the signal in the FOR: Perceptual sensitivity to discriminate signal from noise decreases toward the periphery of the FOR (Fig. 1).

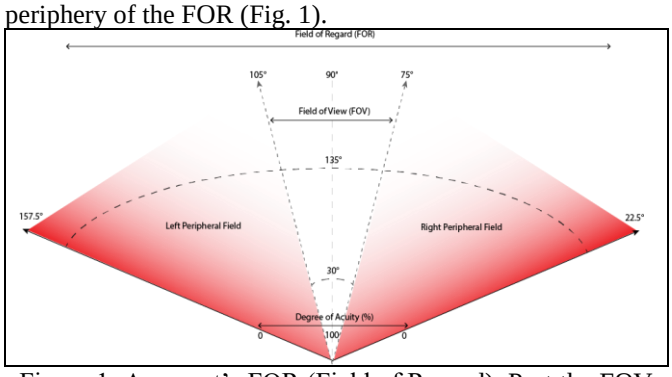


Figure 1. An agent's FOR (Field of Regard). Past the FOV (Field of Vision; 75°-105°), receptor sensitivity declines

with increasing distance from the fovea.

After a signal is detected in the visual field, the agent makes a decision about the identity of the signal, such as if it is a threat or not. The criterion defining the perceptual judgement between noise and objects-of-interest in the environment (e.g. threat vs. non-threat) is given by SDT's utility function: $u(x) = \alpha hP[CD] + \alpha mP[MD] + (1-\alpha)aP[FA] + (1-\alpha)\,jP[CR]$, where $P[\ldots]$ is the probability of each of the four possible outcomes, correct detection (*CD*), missed detection (*MD*), false alarm (*FA*), or correct rejection (*CR*); α is the base rate probability of encountering a signal; $1-\alpha$ is the probability of encountering noise; and *h*, *m*, *a*, and *j* represent the payoffs (benefits or costs) for hits, misses, false alarms, and correct rejections, respectively. Thus, the expected utility of adopting a decision threshold at a particular signal value, *x*, is defined by the probabilities of four outcomes, the base rate, and the payoffs. The optimal decision threshold is found at the point of the highest utility (Fig. 2).

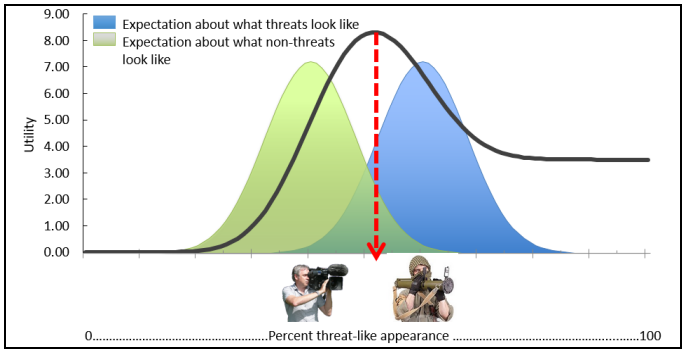


Figure 2. A signals-approach to threat detection. Blue and green Gaussians represent probability densities defining what threats and non-threats look like, respectively, with notional mean appearance of each category depicted by the combatant and journalist. When perceptual uncertainty exists (depicted as overlap of the Gaussians), mistakes cannot be eliminated, but exposure to them can be optimized. The maximum of a utility function (black curve) locates this optimum: the decision criterion that will maximize net benefits over a series of decisions.

Integrating Moderating Variables

Under our model, how effective an agent is at perceptual decision making is dependent on the agent's situation awareness, defined as the accuracy of its signal transduction at the receptors and its estimates of the CD, FA, etc. probabilities; the base rate; and the payoffs. In order to simulate the influence of operational variables on perceptual decision making in Warfighters, we applied a function that affects the accuracy of these parameters as function of moderating variables, such as fatigue.

For example, distortion of the receptor transduction (or another parameter, such as the base rate) can be modeled as a simple sum of the incoming signal value, *x*, plus the influence of behavioral moderators: $x^* = x + Mu(ba)$, where $Mu(ba)$ is the square of either a single or a collection of behavioral moderators. The combined influence of several moderators can be determined by a number different methods, including only using the value of the moderator with the most influence. For this project, we have employed a logarithmic combination method (e.g. Moors, 2009): $ba = 0.1 * \log_2 (\sum sgn(bi) * 2^{10*|b_i|})$, where b_i is an individual moderator in a set of behavioral moderators acting upon an agent. Any moderator variable or collection of variables, then, can affect the perceived value of a signal or effect how that signal is judged by influencing the agent's estimate of the optimal threshold location – a pattern consistent with cognitive affective research (Lynn et al., 2012).

Acknowledgments

This material is based upon work supported by the US Army Command Center, Aberdeen Proving Ground, Natick Contracting Division ACC-APG-NCD under Contract No. W911QY-17-C-0009. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US Army Command Center, Aberdeen Proving Ground, Natick Contracting Division ACC-APG-NCD.

References

Bates, J., et al. (1994). The role of emotion in believable agents. *Communications of the ACM, 37*, 122–125.

Janelle, C. M., & Hatfield, B. D. (2008). Visual attention and brain processes that underlie expert performance: Implications for sport and military psychology. *Military Psychology, 20*, S39.

Loyall, A. B., & Bates, J. (1991). Hap: A reactive, adaptive architecture for agents. Carnegie Mellon University Department of Computer Science Technical Report CMU-CS-91-147.

Lynn, S. K., & Barrett, L. F. (2014). "Utilizing" signal detection theory. *Psychological Science, 25*, 1663–1673.

Lynn, S. K., Zhang, X., & Barrett, L. F. (2012). Affective state influences perception by affecting decision parameters underlying bias and sensitivity. *Emotion, 12*, 726.

Moors, A. (2009). Theories of emotion causation: A review. *Cognition and Emotion, 23*, 625–662.

Schmitz, T. W., De Rosa, E., & Anderson, A. K. (2009). Opposing influences of affective state valence on visual cortical encoding. *Journal of Neuroscience, 29*, 7199-7207.

Staal, M. A. (2004). Stress, cognition, and human performance: A literature review and conceptual framework. NASA Technical Report NASA/TM-2004-212824.

Zacharias, G. L., Miao, A. X., Illgen, C., Yara, J. M., & Siouris, G. M. (1996). SAMPLE: Situation awareness model for pilot in-the-loop evaluation. In *Proceedings of the 1st annual conference on situation awareness in the tactical air environment*.

Similarity-based and Rule-based Reasoning in Raven's Matrices

Can Serif Mekik (mekikc@rpi.edu)
Ron Sun (rsun@rpi.edu)
Department of Cognitive Science
Rensselaer Polytechnic Institute, Troy, NY 12180

David Yun Dai (ydai@albany.edu)
Department of Educational and Counseling Psychology
State University of New York at Albany, Albany, NY 12222

Abstract

Raven's Matrices are a well-known family of intelligence tests. Based on an analysis of a recently proposed similarity-based approach, we present a new analysis of performance in the task. This analysis represents an alternative to the dominant rule induction approach. In particular, it suggests that similarity-based and rule-based reasoning may synergistically enable performance. We discuss possible complementary roles of the two types of reasoning, suggest possible explanations of observed empirical phenomena, and compare and contrast our model with other prominent models of the task.

Keywords: Raven's Matrices; Similarity-based Reasoning; Rule-based Reasoning

Introduction

The *g*-factor is a psychometric construct that reflects general intelligence (Nisbett, et al., 2012). *Raven's Progressive Matrices* (RPM) are among the best single tests for measuring the *g*-factor (Snow, Kylonnen, & Marshalek, 1984). The tests are composed of items that require participants to select, from a set of alternatives, the visual figure that best completes a given array (Raven, Raven, & Court, 1998).

Cognitive modeling research has targeted cognitive mechanisms underlying RPM performance as well as sources of variability in performance (e.g., Carpenter, Just, & Shell, 1990; Lovett & Forbus, 2017). However, models still have much to capture about the cognitive mechanisms humans use to tackle RPM. In previous work (Mekik, Sun, & Dai, 2017), we hypothesized that incorporating feature abstraction processes in models of RPM may place important constraints on processing. In that study, we developed a model tackling a subset of RPM-style problems. Our model used a deep convolutional neural network to model feature abstraction and rule-based reasoning to model strategy selection and/or use.

More recently, we developed a similarity-based reasoning (SBR) approach to RPM-style problems that operates on basic and/or relational visual features (Mekik, Sun, & Dai, 2018). In contrast with our previous work, this new model makes relatively weak assumptions about the structure of RPM items and processes subsymbolic visual representations. We tested various aspects of the SBR approach on 74 *Sandia matrices* (Matzen, et al., 2010), which are computer generated matrix problems that have psychometric characteristics similar to the *Standard Progressive Matrices* edition of RPM. Performance of the similarity-based model was about 85% correct. We believe that the similarity-based approach may lead to a better understanding of human cognitive processing in RPM.

RPM items are viewed in the cognitive modeling and experimental psychology literatures as *rule induction* problems. Thus, cognitive research on RPM focuses primarily on rule-based reasoning (RBR). Yet, much of common-sense reasoning may be captured by a combination of SBR *and* RBR patterns (Sun, 2016; Sun, 1995). Although relatively little attention has been devoted to the possible role of SBR in human RPM performance, verbal overshadowing work (DeShon, Chan, & Weissbein, 1995) suggests that human subjects may use SBR, as well as RBR, when attempting to solve RPM problems. Existing models of RPM do not provide detailed accounts of the possible role of SBR versus RBR in RPM performance. We think that this is an area where our model can offer some useful insights.

Here, we argue for the possibility of a synergistic division of labor (Sun, 2016; Sun, 1995) between SBR and RBR in RPM performance based on an analysis of our SBR approach to RPM and various considerations about computational tractability and human cognitive limitations. We begin with a non-technical presentation of the SBR approach to RPM and then discuss complementary roles for RBR and SBR. We argue that the combined model of SBR and RBR in RPM is qualitatively consistent with several experimental findings and notable cognitive models.

Similarity-Based Approach to RPM

RPM tests are composed of a series of *matrix problems*. Each item presents subjects with a square array of figures (*the matrix*), with the bottom right figure left blank. Subjects are also presented with a second array of alternative figures, and they are instructed to choose the alternative that best completes the matrix. See Figure 1 for an example. RPM tests include both 2×2 and 3×3 matrices, with the latter variety generally being more difficult.

For our purposes, the relevant units of analysis are rows and columns of matrix figures, which we collectively call *figure sequences*. There are two types of figure sequences. *Matrix sequences* are complete matrix rows and columns, and *alternative sequences* are sequences produced when an alternative is inserted into the blank. The *sequence number* is

an index that disambiguates sequences of the same type along the same axis. All indexes are assigned according to standard English reading order (left to right, top to bottom).

RPM item structure is formally specified in Penrose and Raven (1936). This specification implies that each matrix exhibits characteristic row and column features (basic and/or relational visual features). Subjects can find the correct answer to a matrix problem if they can grasp the relevant row and column features and find the figure that produces alternative sequences exhibiting those features. These tasks may be accomplished with basic and/or relational visual features and SBR.

In our SBR approach, basic and/or relational features in matrix and alternative sequences are identified using a neural network. To discover characteristic row features, feature representations of matrix rows are combined by taking the elementwise *geometric mean*[1] of row feature vectors. Characteristic column features are identified in a similar manner. The similarity of alternative sequences to corresponding matrix sequences is then determined using *relative entropy*[2] as a similarity measure. A response is selected according to a *Boltzmann distribution*[3] on the alternative set, with preference for alternatives that minimize the entropy of alternative sequences relative to corresponding matrix sequences. It is important to note that we assume, in relative entropy calculations, that experimental features are mutually independent. Violation of feature independence leads to graceful degradation of performance. Thus, small violations of the independence constraint are permissible.

The implementation discussed in Mekik, Sun and Dai (2018) uses a set of thirteen binary features, though the SBR approach can also accommodate non-binary and continuous features. Basic and relational features used in our tests of the SBR approach were derived from patterns that are used in the computational literature on RPM (e.g., Carpenter, Just, & Shell, 1990). For instance, *increments features* are characteristic of sequences where an attribute is progressively incremented (e.g., shapes get progressively bigger along a row); *distribution features* are characteristic of sequences where distinct values of the same attribute appear in each sequence figure (e.g., same shape but different sizes appear in a row in no particular order).

Walkthrough Example

The SBR approach may tackle Figure 1 as explained below. For this example, only two features are needed: shape distributions and shading distributions. These features are a subset of those used in Mekik, Sun and Dai (2018).

Table 1 shows the output of a perceptual neural network on each figure sequence associated with Figure 1. For instance, the row labeled $mat\ 1$ displays subjective probabilities for shape distribution and shading distribution features in the first matrix row (top) and column (left). In this particular case, the model believes that the first matrix row likely does not exhibit a shape distribution (11% probability that a shape distribution is present) and that it exhibits a shading distribution (99% probability that a shading distribution is present). The network's output on the first matrix column can be interpreted in a similar way: this time, the model believes that a shape distribution is present, but not a shading distribution. When combined with observations from the second matrix row and column using the geometric mean, these observations lead to the conclusion that, overall, the shading distribution feature is characteristic of matrix rows and the shape distribution feature is characteristic of matrix columns (see row $combined$). The remaining rows are the network's observations about alternative sequences.

Relative entropy increases as the information shared by two random variables decreases. Table 2 presents relative entropy similarity indices for each feature in each alternative sequence. For instance, the similarity score obtained by the alternative row generated by alternative 1 (top left) for the shape distribution feature is:

$$.49 \log_2 \frac{.49}{.02} + (1 - .49) \log_2 \frac{(1 - .49)}{(1 - .02)} = 1.81$$

The sum of all similarity indices obtained from a particular alternative, shown in the total column of Table 2, is an overall measure of the similarity of the alternative in question to corresponding matrix sequences, giving equal weight to rows and columns. The alternative that generates the smallest sum is thus the alternative that produces alternative sequences most similar to corresponding matrix sequences (with respect to a current feature set).

In this example, alternative 5 (bottom left) is correctly identified as being the best alternative for completing the matrix, with a total similarity index of .17. In general, responses are selected stochastically, based on the summed similarity indices, according to the Boltzmann distribution. Selection probabilities (assuming a temperature of $\tau = 1$) are given in the Pr column of Table 2. When all alternatives are considered, there is a 76% probability of success on this particular example.

Comparison of Similarity-Based Approach with Existing Rule-Based Models

Since its publication, Carpenter, Just and Shell (1990) has set the agenda for cognitive modeling research on RPM. In this model, matrix problems are analyzed as rule induction problems. The model has a set of rules describing various row-wise patterns and it attempts to match these patterns in matrix rows by executing pairwise figure comparisons. Once

[1] The geometric mean of $x_1, x_2, \ldots, x_n$ is given by $\prod_{i=1}^{n} x_i^{1/n}$.

[2] The entropy of a discrete probability distribution q relative to a distribution p is:

$$D(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}$$

[3] The Boltzmann distribution over n elements is given by

$$\Pr[X = i] = \frac{\exp\left(x_i/\tau\right)}{\sum_{j=1}^{n} \exp\left(x_j/\tau\right)}$$

for temperature τ and activations $x_1, x_2, \ldots, x_n$.

a matching rule set is found, the model selects the alternative predicted by the rule set as its answer.

The Carpenter, Just and Shell (1990) study shed light on the types of patterns human subjects tend to look for, the role of incremental processing, and the sources of working memory load in RPM. To address these issues, some simplifications were made. The model did not attempt to address necessary perceptual processing in detail, though some important constraints were identified (in particular, the problem of *correspondence finding* discussed below). Furthermore, the model had hard-coded rules, and thus focused more on rule recognition than on rule generation. More recent models of RPM have attempted to extend understanding of cognitive processing in RPM by addressing these limitations. A representative example in this regard is the Lovett and Forbus (2017) analogical model.

Lovett and Forbus (2017) present a model of RPM following the structure-mapping approach to analogy. The model extracts patterns of feature variation within matrix rows and then generalizes these patterns in order to inform its response. The extraction process involves restructuring figure representations until a satisfactory *pattern of variance* is found (i.e., *perceptual reorganization*). Then, a structure-mapping process allows the model to identify patterns of variance that are common between matrix rows, yielding generalized patterns that can be used in response selection.

To determine whether a rule applies to a particular row, subjects must make some commitments about which individual visual attributes are subject to the rule; this is the problem of *correspondence finding* (Carpenter, Just, & Shell, 1990). The Lovett and Forbus (2017) model's perceptual reorganization processes address the problem of correspondence finding as well as the problem of rule generation. The model suggests that correspondence finding may be solved perceptually by structurally realigning figure representations until a satisfactory pattern of variance is found. Patterns of variance are essentially rule templates, thus the same process also addresses rule generation in that rules are not hardcoded but are rather systematically constructed out of more basic visual attributes according to patterns of variance.

Our SBR approach agrees with existing models on several important architectural aspects. For instance, the feature set used in our experiments is inspired by the Carpenter, Just and Shell (1990) rule set as mentioned earlier, and it is also comparable to patterns of variance identified by the Lovett and Forbus (2017) model. Furthermore, our approach makes use of row-wise comparisons, in keeping with the aforementioned literature. Finally, our approach involves finding generalized representations of row patterns by combining information from the two matrix rows, just like the models reviewed above. There are some relatively minor differences between our approach and these models. For example, although our use of both column-wise and row-wise processing is somewhat novel, it is an extension of the logic of existing models.

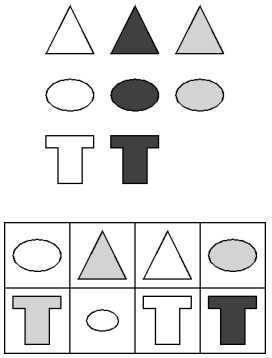


Figure 1: A matrix problem.

Table 1: Subjective probabilities for shape distribution (F_1) and shading distribution (F_2) features on Figure 1 matrix and alternative sequences.

type	seq. num.	row		col.	
		F_1	F_2	F_1	F_2
mat	1	.11	.99	.86	.00
mat	2	.00	.98	.84	.01
$combined$*		.02	.99	.85	.00
alt	1	.49	.47	.55	.52
alt	2	.58	.97	.48	.02
alt	3	.69	.58	.65	.35
alt	4	.34	.91	.54	.03
alt	5	.02	.94	.88	.05
alt	6	.40	.60	.54	.36
alt	7	.02	.40	.89	.54
alt	8	.09	.42	.93	.47

**Geometric mean* of subjective probabilities for matrix sequences.

Table 2: Relative entropy similarity, total similarity, and response selection probabilities for Figure 1.

alt. num.	row		col.		total	Pr
	F_1*	F_2**	F_1*	F_2**		
1	1.81	2.21	.37	3.20	7.59	.00
2	2.38	.01	.53	.03	2.95	.05
3	3.12	1.56	.17	1.90	6.75	.00
4	1.03	.14	.40	.04	1.60	.18
5	.00	.06	.01	.11	.17	.76
6	1.37	1.45	.38	1.94	5.15	.01
7	.00	2.70	.01	3.35	6.06	.00
8	.09	2.54	.04	2.77	5.45	.00

*Shape distribution feature
**Shading distribution feature

Unlike existing models, the SBR approach is not premised on the notion that matrix problems are rule induction problems. This is the fundamental point of difference between our approach and existing models. As a consequence, our approach offers a different perspective on

cognitive and computational challenges presented by RPM (see below). For example, the problems of correspondence finding and rule generation, characteristic of the rule induction approach, do not arise for the SBR approach since the SBR approach relies on basic and/or relational visual features instead of rules governing figure variation.

Role of Rule-Based Reasoning Within the Similarity-Based Approach

The similarity-based approach presents a number of cognitive challenges, some of which are comparable to challenges affecting rule-based models. These challenges may be addressed by a synergistic interaction between SBR and RBR. It is worth mentioning that, even in its current form, the SBR approach relies on some RBR processes, for example, to apply the perceptual neural network to figure sequences, to initiate similarity-judgments, and to set up actual response selection (based on a Boltzmann distribution). However, there needs to be other rule-based mechanisms that orchestrate the similarity-based approach due to the interactive nature and the physical limitations of human cognition.

Consistent with our own previous work (Mekik, Sun, & Dai, 2017) and the literature reviewed above, we believe that RBR handles strategic aspects of RPM. More specifically, we hypothesize that subjects rely primarily on some variant of the similarity-based approach and that the role of RBR in RPM is strategic executive and metacognitive control of various aspects of relevant SBR processes. Below, we discuss and develop some hypotheses about the possible role of RBR in such processes. The combined SBR and RBR model is consistent with the relatively recent Dual-Process Theory of Intelligence, which combines aspects of dual-process theories of cognition with theories of human intelligence (e.g., Sobkow, Traczyk, Kaufman, & Nosal, 2018). The theory distinguishes between implicit (e.g., intuitive) and explicit (e.g., analytic) processes and posits that these processes enable intelligent behavior independently and possibly also through their interaction.

Working Memory Management

As evidenced in Carpenter, Just and Shell (1990), a fundamental challenge in RPM arises from the fact that human working memory capacity (WMC) is limited. Working memory is the store for representing feature probabilities and similarity judgments for the similarity-based approach. Limited WMC implies that subjects can only compare a limited number of features at a time. Our current model simultaneously handles over 10 distinct features at once, which may be stretching human capacity.

The impact of WMC limitations on performance can be minimized through use of memory-efficient features and summarization of intermediate results. Enforcing the use of mutually independent features, as done in the similarity-based model, is one way to decrease the working memory load of the similarity-based approach. Feature independence may be enforced by rules for checking feature dependencies

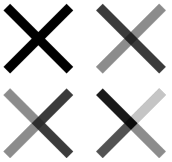


Figure 2: Four different ways to analyze an "X" shape. The shape can be analyzed holistically (top left), as two overlapping diagonal bars (top right), as two overlapping wedges (bottom left), or as four overlapping shorter bars (bottom right) among other possibilities.

and handling dependent feature sets (see below). Independent features minimize redundancies in the information represented by the feature set and simplify computation of similarities by eliminating the need to compute conditional probabilities. Furthermore, relative entropy similarities between figure sequences for two mutually independent feature sets can simply be added to yield the similarity value for the whole set, allowing for easy summarization of intermediate results. Under the assumptions of our similarity-based approach, subjects need only keep track of a running sum of similarity values to incrementally evaluate figure sequence similarities for an arbitrarily large set of mutually independent features. These summarization processes may be controlled by RBR.

Feature Management

A key question for the similarity-based approach, comparable to the rule generation problem for rule induction approaches, is how the feature set is determined. Human subjects likely consider features based on prior general visual and geometric knowledge and abilities, of which the patterns used by our similarity-based approach are a part. But reliance on such a wide base of knowledge, coupled with WMC limitations, results in the need for feature selection and, consequently, the need for feature management processes.

There are two notable aspects of the feature management problem. To focus attention on the first, let us imagine that human visual knowledge is always encoded in terms of basic, mutually independent features. In this case, the feature management problem reduces to a version of the frame problem (Dennett, 1984): the subject must select, from an a priori intractably large set of possibilities, a feature set that is fit for purpose (i.e., one sufficient for accurately evaluating response alternatives). Consequently, exhaustive and systematic approaches seem impractical; feature selection must be driven by heuristic search, which may involve RBR.

The proposition that human visual knowledge is always encoded in terms of basic, mutually independent features is demonstrably false. The same figure can be represented in multiple ways, as shown in Figure 2. In order to enforce feature independence, possible alternative representations of a given figure or pattern must be processed by separate, likely sequential, applications of the similarity-based approach. The task of managing these incompatible representations

constitutes a second notable aspect of the feature management problem. This aspect of the feature management problem is comparable to the correspondence finding problem in rule-based approaches to RPM and may involve perceptual reorganization processes similar to those discussed in Lovett and Forbus (2017). Sequential feature set evaluation and perceptual reorganization may be controlled by RBR.

Strategic Management

The considerations above give rise to problems of a strategic nature. The first problem arises as a direct result of the feature independence assumption. As discussed above, subjects may consider mutually dependent features in different feature sets. In such cases, it is not possible for the similarity-based approach to combine similarities through a simple summation process. Therefore, either feature dependencies must be taken into account, or the model must decide which of several possible feature sets is most appropriate for a given problem. Considerations in this regard may include rules about minimizing the size of the feature set and simultaneously maximizing the discriminative information captured by the feature set.

The second problem arises from the open-ended nature of the feature selection process. Since exhaustive search through the feature space is intractable, additional criteria are necessary for deciding when to stop feature search and similarity assessment processes. Although we believe that motivational variables play an important role here, due to space limitations, we focus only on one possible cognitive explanation. Subjects may evaluate how much variation among matrix figures they have taken into account (Carpenter, Just, & Shell, 1990). A threshold on the amount of variation taken into account, enforced by a set of rules, may serve as a cognitive criterion for evaluating whether sufficient work has been done on a given item.

Further Remarks on the Role of RBR

Given the incremental and heuristic nature of suggested processes and the crisp nature of associated constraints, RBR patterns appear to be the best medium for handling the challenges above. Although SBR may also play a small role, for instance in determining whether two features have a substantial degree of mutual dependence, or in guiding heuristic feature selection. Note that many important RBR processes discussed in this section are generic cognitive processes. For instance, working memory management and feature selection are processes that are likely to be applicable to a wide range of tasks.

It is also worth mentioning that RBR processes may further support RPM performance in other ways. For example, subjects may explicitly abstract patterns that recur in RPM tests and exploit these regularities for improved performance using RBR (Verguts & De Boeck, 2002). Another possibility is that RBR patterns may handle exceptional cases where SBR processes are not readily applicable, for example, due to unavailability of relevant features.

Capturing Empirical Phenomena with the Combined SBR and RBR Model

The empirical literature identifies several cognitive variables affecting human performance on Raven's matrices. One promising feature of the architecture outlined above is that it may capture and explain many of the relevant cognitive effects. Below, we review a few RPM related phenomena and discuss how our model can explain these.

Verbal Overshadowing

DeShon, Chan and Weissbein (1995) divide RPM items into two categories: *visuospatial items* and *verbal-analytic items*. They present evidence that concurrent verbalization inhibits performance in visuospatial items but not in verbal-analytic items, suggesting that at least two processes are involved in RPM performance.

Our model suggests a mechanism that may capture such verbal overshadowing effects. According to our approach, subjects are likely to sequentially consider several distinct feature sets. It is probable that some characteristic features are, on average, considered earlier rather than later, perhaps because they have high visual salience. Likewise, we can imagine the existence of items that can be solved by features that are, on average, considered much later, possibly because they involve uncommon or unfamiliar groupings of elements (and may thus rely more on RBR).

The considerations above suggest the following hypotheses: visuospatial items exhibit characteristic features that are likely to be considered relatively early, whereas verbal-analytic items exhibit characteristic features that are likely to be considered relatively late and with the help of RBR. In visuospatial items, most of the processing should be SBR, and additional RBR processing requirements due to concurrent verbalization may cause interference by diverting cognitive resources. In contrast, concurrent verbalization may not interfere with performance on verbal-analytic items since feature selection in this case may heavily involve RBR.

Generation Speed

Verguts, De Boeck and Maris (2000) present evidence that subjects' rule generation speed is positively correlated with RPM scores. Increased rule generation speed corresponds, in our theory, to increased speed of feature selection and detection processes. We would expect participants with faster feature selection and detection processes to more frequently find appropriate feature sets before they commit to a response, since they would effectively be considering a larger number of features.

Strategic Influence

Vigneau, Caissie, and Bors (2006) investigate individual differences in RPM performance attributable to two strategies. In *constructive matching*, subjects generate possible answer figures and compare these to items in the alternative set. In *response elimination*, subjects consider only the alternatives and try to rule out the ones that do not

fit based on feature comparisons with matrix figures. They find that subjects who prefer constructive matching tend to perform better on RPM tests than subjects who rely on response elimination.

According to our theory, constructive matching may encourage construction of feature sets that more or less capture all variation within matrix sequences since figure construction requires participants to commit to a coherent and complete set of visual features. On the other hand, subjects who employ response matching may not be aware that their understanding of matrix patterns is incomplete or even inconsistent, as they may not have a clear signal for assessing the consistency and completeness of their reasoning. Construction of more complete feature sets should, therefore, generally lead to better performance.

Conclusion

In this paper, we reviewed a model of RPM in the context of important theoretical and empirical considerations. We argued that the similarity-based approach presents a novel perspective on RPM performance as compared to the dominant rule-based approach. In particular, we suggested that both SBR and RBR may be used in RPM performance. We developed qualitative explanations for several notable empirical phenomena using our theory, and we compared and contrasted our approach with two important models of RPM. In future work, we plan to improve the performance of our model and further develop details of SBR and RBR.

Acknowledgments

This work was supported in part by ARI grant W911NF-17-1-0236. Can Mekik was supported by the RPI HASS Graduate Fellowship. We thank Dr. Laura E. Matzen for providing us with the Sandia Matrices and the Sandia Matrix Generation Tool (Matzen, et al., 2010).

References

Carpenter, P. A., Just, M. A., & Shell, P. (1990). What one intelligence test measures: A theoretical account of the processing in the Raven Progressive Matrices test. *Psychological Review, 97*, 404-431.

Dennett, D. (1984). Cognitive wheels: The frame problem of AI. In C. Hookway, *Minds, machines, and evolution* (pp. 129-150). Cambridge University Press.

DeShon, R. P., Chan, D., & Weissbein, D. A. (1995). Verbal overshadowing effects on Raven's Advanced Progressive Matrices: Evidence for multidimensional performance determinants. *Intelligence, 21*, 135-155.

Lovett, A., & Forbus, K. (2017). Modeling visual problem solving as analogical reasoning. *Psychological Review, 124*, 60-90.

Matzen, L. E., Benz, Z. O., Dixon, K. R., Posey, J., Kroger, J. K., & Speed, A. E. (2010). Recreating Raven's: Software for systematically generating large numbers of Raven-like matrix problems with normed properties. *Behavior Research Methods, 42*, 525-541. doi:10.3758/brm.42.2.525

Mekik, C. S., Sun, R., & Dai, D. Y. (2017). Deep learning of Raven's Matrices. *Proceedings of the Fifth Annual Conference on Advances in Cognitive Systems.*

Mekik, C. S., Sun, R., & Dai, D. Y. (2018). Similarity-based reasoning, Raven's Matrices, and general intelligence. *Proceedings of the 27th International Joint Conference on Artificial Intelligence.* Stockholm, Sweden.

Nisbett, R. E., Aronson, J., Blair, C., Dickens, W., Flynn, J., Halpern, D. F., & Turkheimer, E. (2012). Intelligence: New findings and theoretical developments. *American Psychologist, 67*, 130-159.

Penrose, L. S., & Raven, J. C. (1936). A new series of perceptual tests: Preliminary communication. *British Journal of Medical Psychology, 16*, 97-104.

Raven, J., Raven, J. C., & Court, J. H. (1998). *Manual for Raven's Progressive Matrices and Vocabulary Scales: Section 1, general overview* (1998 ed.). Oxford Psychologists Press.

Snow, R. E., Kylonnen, P. C., & Marshalek, B. (1984). The topography of ability and learning correlations. In R. J. Sternberg (Ed.), *Advances in the psychology of human intelligence* (Vol. 2, pp. 47-103). Hillsdale, NJ: Lawrence Erlbaum and Associates.

Sobkow, A., Traczyk, J., Kaufman, S. B., & Nosal, C. (2018). The structure of intuitive abilities and their relationships with intelligence and Openness to Experience. *Intelligence, 67*, 1-10.

Sun, R. (1995). Robust reasoning: Integrating rule-based and similarity-based reasoning. *Artificial Intelligence, 75*, 241-295.

Sun, R. (2016). *Anatomy of the mind: Exploring psychological mechanisms and processes with the Clarion cognitive architecture.* New York: Oxford University Press.

Verguts, T., & De Boeck, P. (2002). The induction of solution rules in Raven's Progressive Matrices test. *European Journal of Cognitive Psychology, 14*(4), 521-547.

Verguts, T., De Boeck, P., & Maris, E. (2000). Generation speed in Raven's Progressive Matrices test. *Intelligence, 27*(4), 329-345.

Vigneau, F., Caissie, A. F., & Bors, D. A. (2006). Eye-movement analysis demonstrates strategic infulence on intelligence. *Intelligence, 34*, 261-272.

A Learning Support System for the Development of Phonological Awareness using a Japanese Word Game

Junya Morita (j-morita@inf.shizuoka.ac.jp)[1],
Junpei Nishikawa (bi16050@s.inf.shizuoka.ac.jp)[1]

[1]Faculty of Informatics, Shizuoka University, 3-5-1 Johoku, Naka-ku, Hamamatsu, Shizuoka 432-8011, Japan

Abstract

Shiritori is a popular Japanese word game for young children. The performance of this game is influenced by factors leading language acquisition, such as phoneme segmentations, correspondence between phonemes and symbols, and access to a large-sized lexical knowledge base. We consider this game to be especially useful for supporting atypical language acquisition, like acquiring a second language, and training people with aphasia and autism. In this research, we present a support system for lexicon acquisition utilizing shiritori, and construct a computational model representing the above factors with the ACT-R cognitive architecture. The model has vocabulary knowledge, limited working memory capacity, and rules extracting phonemes from words. As a result of simulation with a prototype model, some cognitive processes influencing the performance of shiritori became clear. In the future, by improving the model, we can construct a system that monitors the vocabulary acquisitions process for individual users by interacting with the model.

Keywords: Phonological Awareness; ACT-R

Introduction

Language acquisition support is increasingly required in modern society. One background is of internationalization and informationization progress. Opportunities for collaborating with people with a different first language and the need to use a second language have increased. Another context is attention to higher cerebral dysfunction that makes the use of language difficult. Due to the progress in cognitive science research, causes and countermeasures of acquired aphasia and congenital disorders such as autism have become better understood.

Acquisition of a second language and acquisition of a language by a person with a cerebral function disorder are different in many respects from a typical acquisition process of a first language. Normally, a human infant learns a language by observing and imitating the behavior of its caregiver (Tomasello, 2008). In this acquisition process, infants modulate enormous parameters such as patterns of sound segmentation and correspondences between symbols and objects. This adjustment process is induced by innate cognitive constraints such as joint attention and various cognitive biases. However, it is usually difficult to effectively utilize innate constraints in the acquisition of second language and acquisition of cerebral dysfunction language (Baron-Cohen, 1997). For this reason, a person has to exert considerable effort to acquire the language skills equivalent to first language speakers.

In order to support such difficult learning, a learner model corresponding to the individual is necessary. Such models, based on cognitive science, are useful to distinguish the cause behind several stumbling blocks for language learning. Appropriate intervention can be realized by referring to the model in which each cause is separated in the learning support system.

Therefore, this research proposes a method of modeling the vocabulary of individuals who have difficulty in acquiring languages. The features of the method proposed in this research are (1) to use interactive games related to languages, and (2) to estimate the knowledge and parameters of cognitive architecture based on feedback receive from individuals. As the interactive game, we use *shiritori*, which is a Japanese word game, and for the cognitive architecture we use ACT-R (Adaptive Control of Thought-Rational; Anderson, 2007).

Task

Shiritori is a common word game in Japan. Normally, Japanese children start playing this game around the age of four. In the game, players are required to say a word that begins with the final letter of the previous word. For example, after a player answers "りんご" (Roman character: ringo; meaning: apple), the other player continues with "ごりら" (Roman character: gorira; meaning gorilla). If a player repeats a word that has been already used in the game or a word that has a specific end character, the player loses. This procedure is often carried out by several players, but some vocabulary learning materials for young children allow them to play shiritori as a single-player game.

The game is connected to Japanese language characters. Japanese has multiple letter systems, such as kanji and kana characters. Among them, kana characters, including hiragana and katakana, correspond to the Japanese phonological system. Unlike English syllables, Japanese phonemes are unitized by mora consisting of a combination of a vowel and a consonant. Kana characters in Japanese are directly connected with mora. In other words, shiritori is a relation of letters and phonology at the same time.

In Japanese speech therapy, such as the treatment of aphasia and autism, shiritori is frequently used (Haradiyanti, 2014). Several Japanese papers show that the interaction through shiritori is used for examining children with autism during treatment. The conditions that enable shiritori are clarified by a cross-sectional survey targeting children with typical development (Takahashi, 1997). The research indicated that phonological awareness, which divides sound into phonemes and gives phonological index to mental lexicon, is

necessary to make children understand the rules of shiritori. Furthermore, it is shown that the acquisition of kana characters is effective for indexing vocabulary by phoneme, and helpful to execute a shiritori game.

In this research, we aim to model the above conditions that enable shiritori on the cognitive architecture. By incorporating such models into the system, we aim to support acquisition of languages in difficult situations while monitoring the state of learners.

Model and System

The cognitive architecture is the basis for modeling cognitive processes that occur in individual tasks. By using a model with a cognitive architecture, it is possible to construct a model that separates various factors required for achieving the task. Out of the various cognitive architectures, we will focus on ACT-R in this research.

ACT-R has already been used in many research studies on language acquisition. A model related to acquisition of irregular verbs in English (Taatgen & Anderson, 2002) and a model of reference learning by children (Van Rij, Van Rij, & Hendriks, 2010) have been constructed. Studies on brain dysfunction have also been developed, and some studies have described errors caused by aphasia in sentence comprehension using the parameters of ACT-R (Matzig, Vasishth, Engelmann, & Caplan, 2017).

However, previous studies on cognitive architecture have only dealt with English, and there is no research on vocabulary acquisition with shiritori. As mentioned, shiritori is a language task involving phonological awareness. Therefore, this might be a useful subject to explore the correlation between spoken and written language. There is also a possibility that the module structure by ACT-R effectively models such interaction between modalities.

To support language learning through shiritori, the study proposes a system that includes interaction between a human user and an ACT-R model (Figure 1). In this system, a learner interacts with the ACT-R model via typing or voice. The speech interface is generally difficult because it requires precision of speech synthesis and speech recognition. However, in a learning setting to acquire phonological awareness, a conscious effort for input might be effective. Also, by combining speech interface and kana character input, there is a possibility of providing support to learners who find it difficult to obtain phonological awareness.

In the ACT-R, with the visual and manual modules, the interaction through the typing interface is realized. Similarly, with the auditory and voice modules, interaction through the voice interface is realized. The input and output in each modality is integrated in the production module and interact with modules inside the system (goal module, imaginal module, and declarative module). Among the internal modules, the goal module and the imaginal module hold the current situational problem of short-term memory. In contrast, declarative modules hold vocabulary knowledge and the learner's

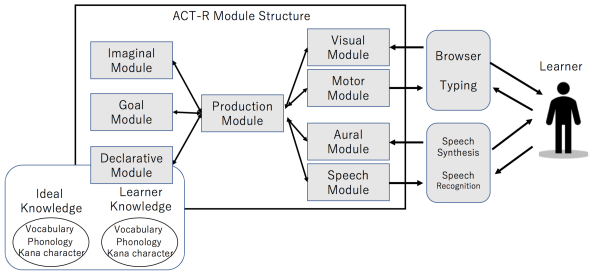


Figure 1: Proposed System.

vocabulary models. The system estimates the latter knowledge through interaction with the learner.

Summary and Future work

In this abstract, we present a system utilizing ACT-R to support atypical language acquisition. Our system focuses on non-English language. We believe that our approach is helpful for fundamentally understanding human cognition and language, especially with regard to spoken and written language interactions.

We have so far developed a prototype model that involves playing a game of shiritori, exploring cognitive factors involved in this task through preliminary simulations. In future studies, we will develop interfaces presented in Figure 1, and conduct psychological experiments where participants can interact with the system.

References

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

Baron-Cohen, S. (1997). *Mindblindness: An Essay on Autism and Theory of Mind*. The MIT Press.

Haradiyanti, C. P. (2014). The study of Aphasia in Megumi's character in the third episode of "Mr.Brain Dorama". *Journal Ilmiaha Mahasiwa Fib*, *2*.

Matzig, P., Vasishth, S., Engelmann, F., & Caplan, D. (2017). A computational investigation of sources of variability in sentence comprehension difficulty in aphasia. In (p. 1-6).

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

Takahashi, N. (1997). A Developmental Study of Wordplay in Preschool Children. *The Japanese Journal of Developmental Psychology*, *8*, 42-52. (In Japanese)

Tomasello, M. (2008). *Origins of human communication*. The MIT Press.

Van Rij, J., Van Rij, H., & Hendriks, P. (2010). Cognitive architectures and language acquisition: A case study in pronoun comprehension. *Journal of Child Language*, *37*, 731-766.

A Computational Model of Sensemaking in a Hurricane Prediction Task

Shane T. Mueller (shanem@mtu.edu)
Department of Cognitive and Learning Sciences
Michigan Technological University
Houghton, MI 49931 USA

Brittany Nelson (bnelson1@mtu.edu)
Department of Cognitive and Learning Sciences
Michigan Technological University
Houghton, MI 49931 USA

Abstract

Sensemaking is described as how people make sense out of the world, and is an emergent process involving the interaction of low-level cognitive functions that have often been studied in isolation. If individuals who perform well in one aspect of sensemaking also excel in other aspects, this suggests it may be valuable to study sensemaking as an emergent coherent process. We discuss an experiment where participants learned to estimate and detect errors based on weather reports. Results showed that systematic individual variability in estimation predicted error detection ability. We then describe a computational model of sensemaking that assumes weights in a predictive model are encoded as a fuzzy ensemble of values that get updated independently via a delta learning rule, and are used to make predictions and detect errors. Two simulation models show that although either learning rate or estimated priors produce reasonable accounts of the data, both are important.
Keywords: sensemaking; prediction; forecasting; learning; error detection

Sensemaking has been described as the ability to make sense of our experiences in the world (Klein, Moon, & Hoffman, 2006a). As such, is a *macrocognitive* process (Klein, Moon, & Hoffman, 2006b) that involves interplay between a number of lower-level processes in service of a ill-specified goal, carried out to accomplish complex behavior, gain better understanding, and performing intelligently in context.

In general, we can consider sensemaking to involve a number of abilities that have been studied in isolation:

- Learning (function learning, category learning)
- Estimation, prediction and judgment, and forecasting
- Forecasting
- Choice between options and decision making
- Detecting anomaly and error
- Causal reasoning
- Problem detection
- Problem solving

These different processes are typically studied in isolation, without considering either the ways the different functions interact, or how they use a common body of knowledge to accomplish a more complex goals. We expect them to work together on the same body of knowledge, which might generically be called a *mental model*, and has specifically has been referred to as a frame in the Data/Frame theory of sensemaking (Klein, Moon & Hoffman, 2006b).

Because of its complexity, the holistic process of sensemaking is rarely studied. One might wonder whether there is value in it at all, as a scientific reductionist approach naturally lends itself to studying the component functions in isolation. One reason to attempt to understand the emergent process is that naturalistic and complex environments and situations, explanations of behavior based on sensemaking are prevalent and in fact typical. This in fact has been the focus of much of the naturalistic and macrocognitive research on sensemaking in individuals, teams, and organizations (see Klein et al.,2006a). We might also consider the core knowledge of sensemaking (i.e., the mental model), and test whether, at minimum, a common body of knowledge is used across related subtasks of sensemaking. Once this minimum criteria is reached, it then may be fruitful to understand how these functions work together, what the most likely sources of knowledge and performance differences are, and how the knowledge gained in one task can be applied to more complex situations. This is the approach we will take in the present work.

Sensemaking in a hurricane prediction task

In cue-learning and categorization literature, many similar paradigms have been used to examine how people predict outcomes based on probabilistic signals. One prominent paradigm is the weather prediction task (WPT) which has been used extensively by Gluck and colleagues (Gluck & Bower, 1988; Knowlton et al., 1994; Gluck et al., 2002) to study learning of probabilistic outcomes (sunny or cloudy) based on a sets of otherwise meaningless cues. Results in such tasks reveal learning trajectories, and also different strategies used (such as considering just the best feature, weighing all features, or considering only positive or negative features, see Gluck et al., 2002). The cognition involved in true weather forecasting is much more complex (see Hoffman, 2017), but at its core weather prediction spans both laboratory and naturalistic contexts, and so is a reasonable testing ground for understanding sensemaking.

We have designed a testing system that is similar to the WPT, with a few exceptions that make it more amenable to studying sensemaking, implemented via the cross-platform experiment-design system PEBL (Mueller & Piper, 2014). Describing the general task, (see Figure 1), participants are

first given a description of a set of eight features that serve as predictors of the risk of a hurricane (bottom left panel). One each trial, they are shown a sensor report describing the state of between 0 and 8 weather variables (top panel), that include status of wind, rain, sea level, and the like. They are told that these indicators have strengths between very good and none, and in truth they combine in a logistic model to predict the probability of a hurricane with feature weights .8, .8, .6, .5, .5, .2, .1, and .01. Next, participants are asked to estimate the probability of a hurricane occurring (bottom right panel), and given feedback, awarding points if their estimate was within 7.5% of the true probability. On some trials, correct feedback was given, but on the rest of the trials, erroneous feedback is given, inducing a true or simulated error. When an error was induced or committed, the participant was then asked whether they felt the system was to blame, or they themselves were. This error detection is a simple way of determining whether their knowledge, as they may be correct even when they guess, but may not be able to determine if an error was produced by the system. This mode of operation also mimics many behaviors of humans operating intelligent systems–they need to trust the system, and their ability to predict the system can help produce trust. Overall, we conceive of the task as having the participant learn a forecasting model, rather than learning the weather–they are told that a computer model is creating the prediction that they are trying to match. The underlying model is a logistic regression model, with different independent features having different weights.

This task involves several opportunities to examine complementary aspects of sensemaking. These include:

- The ability to use instruction to start with a reasonable frame for making a probability prediction
- The ability to use feedback to learn how to combine evidence to make a prediction.
- the ability to identify the source of errors (themselves or a computer model).

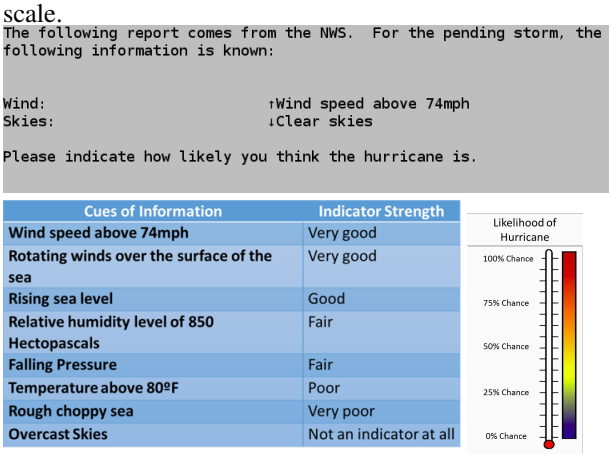
In subsequent and prior studies not discussed here, we have also examined other aspects of sensemaking processes and representations, making evacuation decisions, learning non-independent features, diagnosing sensor errors, and correcting the system to match a given output.

Our goal is to use this task to (1) test whether different sensemaking functions rely on a common mental model or knowledge base; and (2) identify possible sources for this knowledge in a computational model that might explain individual differences in performance.

Method

All methods were approved by the MTU institutional review board (IRB). We tested 27 participants who took part in the study in exchange for course credit. Participants were first instructed in the task, given several practice trials, and then allowed to move on to the testing phase. They completed a total of 80 trials, which involved combinations of up to 8 features. On each trial, they first made a probability estimate. Next,

Figure 1: Elements of hurricane prediction experiment. Top panel shows typical message. Bottom left shows feature description, and bottom right shows likelihood assessment scale.



the system determined whether an error would be signaled. Errors were signaled whenever the participant made a true error, and on half of all other trials. On system-correct trials, the true probability of the model was shown; on system-error trials a new probability was sampled that was outside a 7.5% region around the true estimate. Participants were asked to determine whether the error came because of themselves ("I was to blame") or if they thought the system made an error ("the system was to blame"). All participants experienced the same 80 trials, but in a randomized order, and the trials were sampled to produce a range of probability estimates. Once they completed the task, they were debriefed and given credit for completing the study.

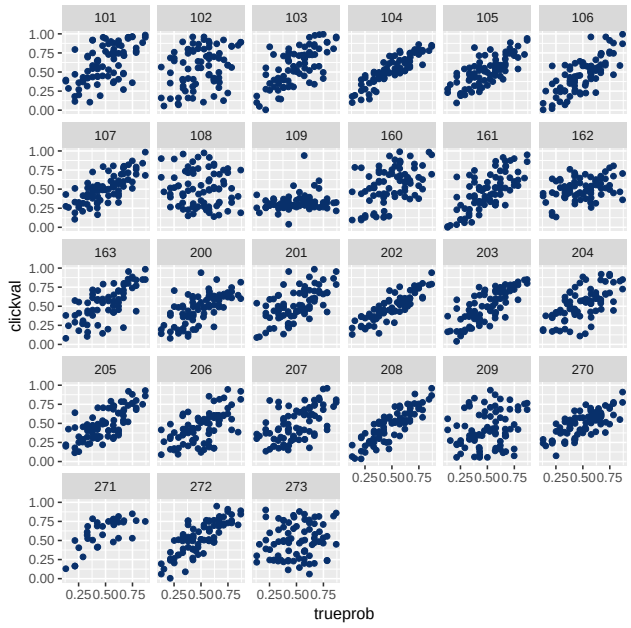
Results

We first would like to establish that performance differs across individuals. Figure 2 shows scatterplots of the given-to-estimated probability across all individuals. The correlations ranged from -.08 to .86, with substantial variability. Our main hypothesis in this study is that the ability to estimate the probability will have related effects in other sensemaking functions, and so we would expect this correlation to predict other behaviors. An ANOVA model incorporating participant code and trial pattern on prediction error showed that participant code was highly significant ($F(26, 1996) = 14.3$, $p < .001$), suggesting that substantial systematic variability stems from individual differences.

As an alternate measure of individual accuracy, we computed the proportion of trials on which a participant was within 7.5 % of the true model value (which is the criterion for being awarded points). These values are seen in Figure 3, arranged by three error conditions, depending on whether the system or the human was correct.

The horizontal axis of Figure 3 shows that the range of abilities in estimation have an impact on blame assessment. The leftmost panel shows that all participants–regardless of

Figure 2: Performance across individuals ranged from close to random (participant 109; R=-.08) to very precise (participant 202; R=.86).



how well they estimate probability, tend to blame themselves when the system was correct but an error was made. The rightmost panel shows what happens when the person is actually correct but the system is incorrect. Here, those who do worse also tend to blame themselves. In contrast, those who do better tend to blame the system. In this case, blaming the system is the "correct" response, and we see that those who are better at estimating the probability are better at assessing blame for errors. The center panel shows what happens when both the human and computer are wrong, and the results do not differ substantially from the rightmost panel. This result demonstrates that ability to estimate the probability is useful for detecting an error and appropriately assigning blame– suggesting that an emergent process is involved.

A Computational Model of Sensemaking

We have designed the experiment so that the mental model of performance is fairly simple, and maps onto a simple linear model with a logistic transform. More complex mental models may require networks or hierarchies of such models, but at its core the notion is akin to modeling estimation with an improper linear model (Dawes, 1979). This is a generalization of the framework was used by Mueller (2009) to implement a core Recognition-primed decision process.

Probability Prediction

The first function to consider is how a probability estimate is made when a set of features is shown. There are numerous theories about how people make estimates in light of complex probabilistic data. Gluck et al., (2002) argued that in a similar context many participants considered only the best features, but others considered more. Our own intuition is that we perform some intuitive mental arithmetic: starting at a baseline value (e.g., 50%) and adjusting the probability higher and lower based on the presence of positive and negative features, assuring to give the stronger indicators more weight and not permit probabilities to extend outside the 0-100% range. In post-experiment interviews, we have found that participants have sometimes reported strategies like this, but they also indicate many misconceptions, including setting a baseline that is not 50%. So, our initial model assumes that for each predictor, people understand its relative weight in predicting a probability of a storm, these weights add together to form an internal odds ratio, which is transformed to a probability using a logit transform $logistic(x) = 1/(1+e^{-x})$. This is identical to the logistic regression model without error:

$$p = logistic(\sum_i f_i \beta_i) \quad (1)$$

Here, f_i are feature values that are either +1 (consistent with hurricane), -1 (consistent with calm), or 0 (absent). The coefficients β_i can differ for each predictor, but in our initial model, all features that are present are used, and a probability is estimated exactly from the beta weights and feature values, with no error. If an individual has a good intuitive estimate of the β_i values, they should be able to produce a close approximation to the true probability; otherwise, they are likely to mis-estimate the probability.

Learning rule

. Next, we must consider how the β_i weights change over time. Many learning schemes are based on the so-called 'delta' rule: if a feature is present and its value is positive, errors between the model and the feedback signal should adjust the beta weights to reduce the likelihood of future error (see Rescorla & Wagner, 1972; Gluck & Bower, 1988). If the message contains features $f_i \in -1, 0, 1$, the weights at time j are $\beta_i^{[j]}$, the difference between the estimated probability and the feedback is ε, and α is a learning rate parameter,

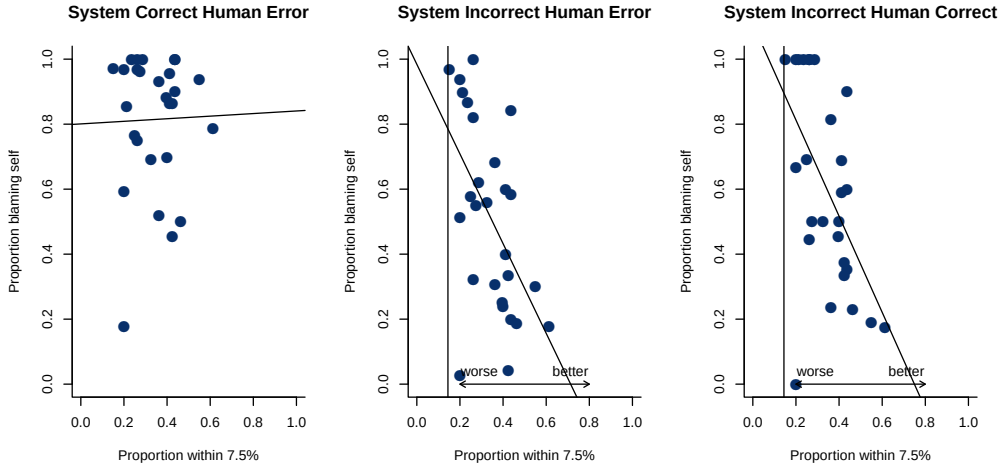
$$\beta_i^{[j+1]} = \beta_i^{[j]} + \varepsilon * (1 - f_i) * |f_i| * \alpha \quad (2)$$

The inner $1 - f_i$ implicitly assumes that the canonical pattern for maximum chance of a hurricane is all features set to 1.0. For more complex classification schemes in which different classification patterns are of interest, this could be replaced with $p_i - f_i$, where p_i is a prototype classification pattern. The logic of the learning rule is that β values are changed from previous values in the direction that would reduce error, but only for the features that are present (-1 or 1).

Error and Anomaly Detection

As currently described, it is not clear how an error could reasonably assessed. This is in part because the model has no

Figure 3: Results from weather sensemaking experiment. Individuals who are better at estimating the model's probability are also better at discriminating correct model behavior (left panel) from incorrect model behavior (center and right panels).



sense of how sure it is about its β values. For any set of features, it can provide an estimate, but this estimate will almost never be exactly the same as the true model prediction. Thus, we need to consider how the model knows the value is likely to have arisen from its internal model.

Clearly, a number of mechanisms could suffice. A simple bounded confidence model would identify any estimates greater than some fixed difference as outside the model's bounds, but Mueller and Tan (2017) questioned the utility of such models in many contexts. Many likelihood-based model and fuzzy models could have the ability to provide bounds on the outcome estimates. For example, if the β weights were not point estimates but rather distributions, the posterior estimate would also be a distribution, and the likelihood of that distribution could be measured by applying Bayes rule–and in fact the β weights could be estimated via Bayesian inference.

We have opted for an approach that approximates this, without requiring the mechanisms and distributional assumptions of Bayesian inference. We assume that instead of a single set of *beta* weights, the individual has a collection of independent prediction models whose coefficients start randomly and evolve independently over time. One might consider the multiple versions of each parameter an empirical prior distribution, or a fuzzy-set representation of the uncertainty in the value. For example, an agent may have have ten versions of beta estimates. On each trial, a prediction can be made by sampling one or more of these versions and using it to compute estimates. This provides both a noisy estimation, and the possibility of using mental simulation to sample a range of possible values. When feedback is given, a subset of the models (as few as 1, but multiple are possible) are sampled and updated using Equation 2. Models thus can start out with a wide range of values, producing a range of possible outcomes, but over time will converge to similar weights and similar predictions. Thus an agent that has multiple sets that are similar can be highly confident in its estimate (even if it is not calibrated), whereas an agent with very different weights will have less confidence in its estimate.

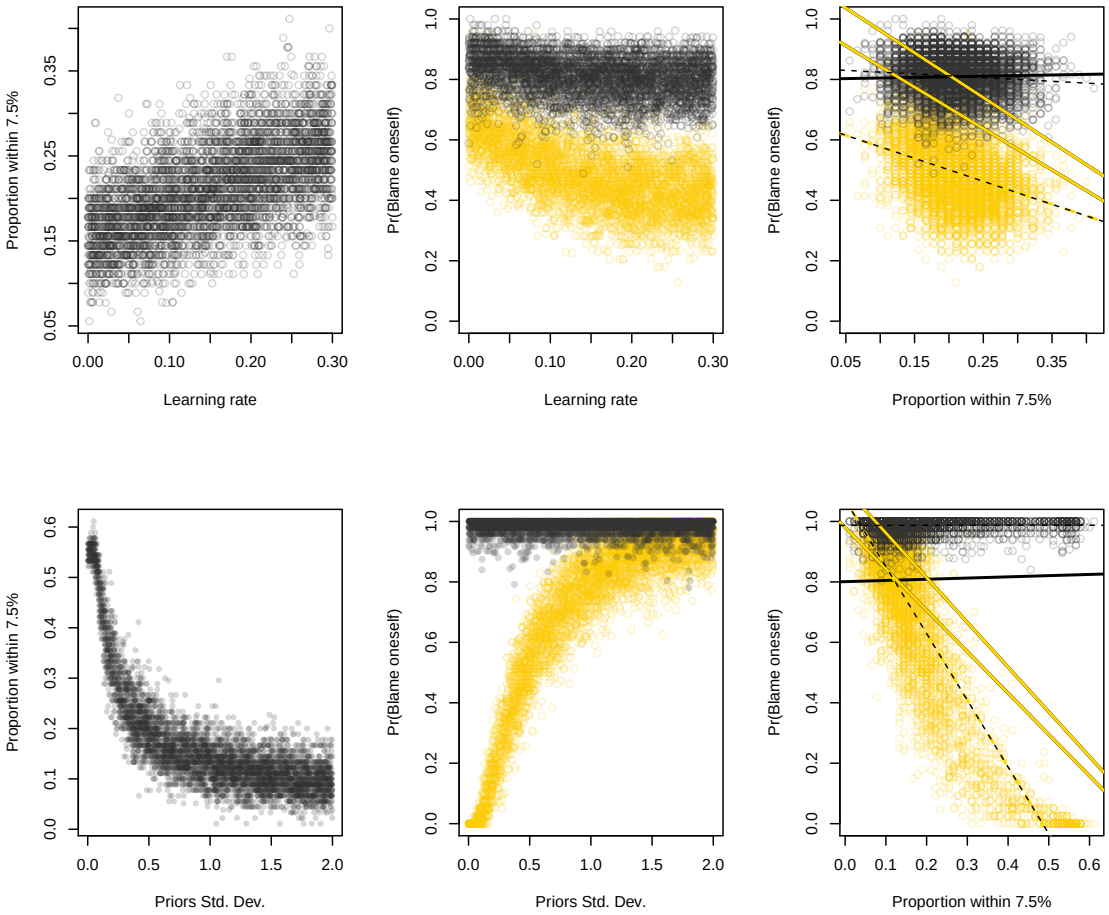
This scheme has a direct functionality for blame assessment. An agent can make an estimate by sampling one model and producing an estimate. But if the agent is given another estimate, it can estimate the likelihood that the estimate arose from any of its models. Although this posterior distribution could be generated by smoothing or estimating a specific distribution, we will use the simple decision policy of asking whether the given estimate is outside the entire range of estimates the agent could make at any point in time. An estimate outside the set of possible estimates is considered anomalous or an error.

In the current design, detecting an error is synomomous with assigning blame. That is, if the given feedback is outside of the agent's bounds, it is judged as being the fault of the system, rather than the agent. Thus, agents with β weights that are similar will be likely to detect that the system is broken even when the amount of error is small, but agents with widely varied weights are likely to produce an estimate even more extreme than the error, and thus attribute the error to their own lack of knowledge.

Simulation

The behavioral data showed systematic differences in how accurate people were in estimating the probability. There are multiple ways in which we might represent individual differences in ability in the model. Two natural explanations are that (1) there are differences in how well people learn in response to feedback; and (2) there are differences in how well people generate initial (prior) estimates of β weights in response to instructions. There are other possibilities as well, such as how many versions of the β weights are stored, how many are update in response to a given piece of feedback, and

Figure 4: Results from two simulations (n=5000). Left panels show how prediction error is impacted by varying either learning rate (top panel) or prior noisy (bottom panel). Center panels show impact of learning rate on probability of blaming oneself either when the system is correct (black) or incorrect (gold). Right panels show relationship between prediction accuracy and blame, analogous to data in Figure 2. Dashed lines represent best-fit linear-model for simulation, and solid lines represent best-fit linear model from human data in Figure 2.



the extent to which multiple versions might be sampled prior to making an estimate.

To examine how this model can account for results of the experiment, we conducted two simulation studies. In each study, we had each agent go through the 80-trial design that human subjects were exposed to, making an estimate, getting feedback, and assigning blame on each trial. For each of 5000 simulated participants, we recorded their accuracy across the 80 trials (in terms of the proportion of trials within 7.5% of the true value), and their blame assessment probabilities for both system correct and incorrect trials (mapping onto the left and right panels of Figure 3). In all models, agents initiated an ensemble of 8 beta-weight sets, and on each update, four sets were sampled (with replacement) and a single delta-rule step was initiated on each. These simulations have no free parameters for fitting the human data in Figure 3, and the only arbitrary parameters involve the number of beta-weight ensembles, the number of ensembles sampled on each set, along with the prior distribution in Simulation 1 (set to a uniform distribution between 0 and 2) and the learning rate in Simulation 2 (set to 0).

The top panel of Figure 4 shows simulation results for a model in which the main between-agent variable was the learning rate. On each simulation, a learning rate parameter was sampled between 0.0 and 0.3 (horizontal axis of left panel of Figure 4). As learning rate increased, overall accuracy increased (note that as seen in Figure 3, typical values produced by human participants ranged from about .2 to .6). As learning rate gets larger, we see that the ability to discriminate system-correct errors (black points) from system-incorrect errors (gold points) increases. The right panel shows a plot akin to that in Figure 3. Overall, the system-error trials show very similar results to the human data. In contrast, the system-incorrect condition shows a negative slop (like human data

did), but there are some overall mismatches. Nevertheless, the qualative pattern is similar, suggesting that a learning-rate account may explain the individual differences we found.

The bottom panel of Figure 4 shows a model that involves no learning ($\alpha = 0.0$), but systematically varies the standard deviation of the different examplars of each β weight around a starting level that matched the true model (.8, .8, .6, .5, .5, .2, .1, and .01). The rightmost panel shows that as the standard deviation of the values increased, overall accuracy decreased, and when the standard deviation was near 0, performance was close to 50% (half of trials were estimated within 7.5% of the true value). Now, when comparing performance to blame assessment, the slope of the system-incorrect trials (in gold) is perhaps a better approximation of the human data. The main problem is that when the system is correct, the agent correctly blames the system close to 100% of the time, in contrast to humans who blame the system around 80% of the time. This difference likely arises because the starting values were centered on the true values, and so the correct estimate would almost always be in the center of the estimates. Had the starting mean beta weights been less calibrated, more correct estimates would have been outside the agents estimates and this probability would have been lower.

Discussion

The two simulations provide two examples of model parameters that might produce the systematic variability we saw in our human experiments. Each produced some mismatches to data, but they are nevertheless informative about what kinds of individual differences may produce systematic differences in sensemaking ability. We find the model involving better assessment of priors more satisfying in its account of our data, aside from it being *too good* when it made an error.

To examine the difference between these in greater detail, we fit a linear regression model predicting overall error rate by participant and trial, and also computed accuracy across the entire experiment and for the first five trials.

We found that the accuracy for the first five trials was highly correlated with the performance over the entire task ($R = .79$, $t = 6.4$, $p < .001$). Performance on the first five trials was negatively correlated with slope ($R = -.41$, $t = 2.23$, $p = .03$), but performance over the entire experiment was not ($R = -.15$, $t = .76$, $p = .45$). The negative slopes indicate that people who had less error at the beginning overall had relatively more positive slopes. In other words, those who started out good improved less than those who started out more poorly. This suggests that both of the factors studied in the simulation may be at play. Some people are able to set good priors, and so if they are, they are good immediately and end up doing the best of all participants. Others start out less good, and tend to improve more, but their overall performance remains worse than those who start off good.

Conclusions

This model attempts to integrate several complementary functions related to sensemaking that have previously been studied mainly in isolation. We found that performance in one skill (estimation) is predictive of performance in a second task (assessing whether an error arose because of the person or the system). Our computational model of sensemaking integrates different functions via a simple model, and incorporates error detection by assuming that feature-weights are fuzzy and encoded as an ensemble of values that get updated independently. This type of assumption is not necessary when studying just the learning aspect of the task, but by adding additional sensemaking tasks, we are able to identify, simulate, and discriminate sources of individual differences, which involve both ability to set reasonable feature-weight estimates based on instruction, and ability to learn based on trial-by-trial feedback.

Acknowledgments

The experiment presented in this paper was part of the Master's thesis of BN. The modeling was supported by DARPA Explainable Artificial Intelligence (XAI) program.

References

Dawes, R. M. (1979). The robust beauty of improper linear models in decision making. American psychologist, 34(7), 571–582.

Gluck, M. A., & Bower, G. H. (1988). From conditioning to category learning: An adaptive network model. Journal of Experimental Psychology: General, 117(3), 227–247.

Gluck, M. A., Shohamy, D., & Myers, C. (2002). How do people solve the "weather prediction" task?: Individual variability in strategies for probabilistic category learning. Learning & Memory, 9(6), 408–418.

Hoffman, R. R., LaDue, D. S., Trafton, J. G., Mogil, H. M., & Roebber, P. J. (2017). Minding the weather: How expert forecasters think. MIT Press.

Knowlton, B. J., Squire, L. R., & Gluck, M. A. (1994). Probabilistic classification learning in amnesia. Learning & Memory, 1(2), 106–120.

Klein, G., Moon, B., & Hoffman, R. R. (2006). Making sense of sensemaking 1: Alternative perspectives. IEEE intelligent systems, 21(4), 70–73.

Klein, G., Moon, B., & Hoffman, R. R. (2006). Making sense of sensemaking 2: A macrocognitive model. IEEE Intelligent systems, 21(5), 88–92.

Mueller, S. T. (2009). A Bayesian recognitional decision model. Journal of Cognitive Engineering and Decision Making, 3(2), 111–130.

Mueller, S. T., & Piper, B. J. (2014). The psychology experiment building language (PEBL) and PEBL test battery. Journal of neuroscience methods, 222, 250-259.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. Classical conditioning II: Current research and theory, 2, 64–99.

Predicting Learning and Retention of a Complex Task

[1]Jacob D. Oury (OuryJacob@psu.edu)
[2]Farnaz Tehranchi (farnaz.tehranchi@psu.edu)
[1]Frank E. Ritter (frank.ritter@psu.edu)
[1]College of Information Sciences and Technology, Penn State, University Park, PA 16802 USA
[2]Department of Computer Science and Engineering Penn State, University Park, PA 16802 USA

Abstract

We use an ACT-R model of a complex task to explore the implications of ACT-R's learning and forgetting mechanisms to better understand learning and retention. The model performs a task that has 14 non-iterated subtasks that takes approximately 25 min. to perform the first time. The results show that a typical learning curve is generated by the model that is well fit to human data. When decay is examined we find that the retention curves basically match the shapes predicted by the KRK theory, and that training and testing have been confounded in many studies. From these results we see that the previously hypothesized mixed declarative procedural stage of learning actually starts on the first trial and is never completely exited, so we will need to propose other thresholds to mark transitions between declarative, mixed, and proceduralized knowledge. We predict based on this model that learning and retention will vary greatly by task components, practice schedule, and learner's strategy.

Keywords: ACT-R; learning curve; retention model

Introduction

Kim, Ritter, and Koubek (2013) provided a summary of learning theories in a review paper. Their summary theory is based on learning theories by Fitts (1964), Anderson (1982), Rasmussen (1986), and VanLehn (1996). It is also consistent with further work reviewed by Kim et al. (2013), as well as other theories of learning (e.g., Posner, 1973) and data on learning (e.g., Seibel, 1963)–the diagrammatic representation of this theory is shown in Figure 1.

In this paper we first briefly review the work this theory is based upon to suggest how to better test it. We then use the ACT-R cognitive architecture to make predictions about learning and forgetting, including where the stages might appear. We then test the predictions of the theory using data (Kim & Ritter, 2015), and use the model to design a further empirical study to test and illustrate the theory's predictions.

The KRK Theory

This review starts with the KRK theory and its predictions. Some of the data and theories that it was based on and some further work, both empirical and theoretical is examined to find further support and limitations. Implications for further development are provided in a summary.

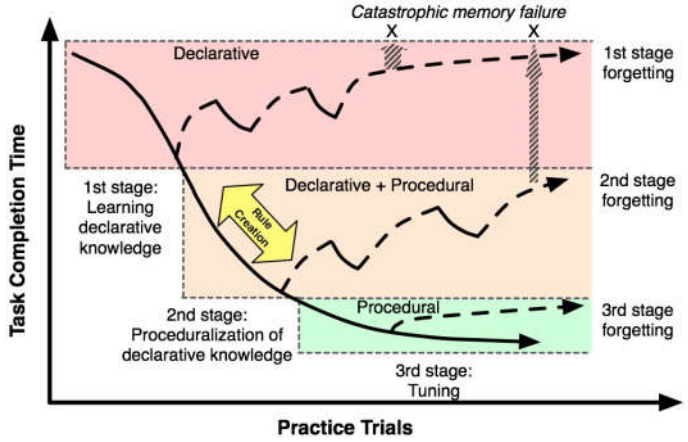


Figure 1. The KRK theory of learning. This represents a summary theory of predictions 1-5 from Table 1. Taken from Kim, Ritter, and Koubek (2013).

The KRK theory was developed as part of Kim's PhD thesis (Kim, 2008). The theory, shown in Figure 1, makes several predictions implicitly. Table 1 explicitly shows predictions from Figure 1 and new predictions. In a larger paper we explain these predictions in more detail (Ritter et al., forthcoming). Here, we simply summarize them before testing them.

Of the nine hypotheses in Table 1, items 1-6 are basically supported in the Kim et al. (2013) review paper. Items 7-9 are new and come from the inclusion of perceptual-motor and recognition memory. The empirical support for the hypotheses are piece-wise and often on simple tasks (e.g., Choice Reaction Theory, Seibel, 1963). The forgetting curves often are from single points of learning (e.g., Kim 2008). That is, we do not know of a single study that predicts all these curves, and most of the studies that are used to derive and support these hypotheses use simple tasks, such as word association.

It would be useful to explore these predictions with a complex, multi-step task and to explore the whole set of predictions with a single empirical study with longer retention intervals. This study would be a large undertaking. So we will use a model of a complex task in an architecture with learning and forgetting that has multiple skill representations to explore the study first and see what the model's predictions are for learning and retention.

Table 1. Human performance hypotheses from the KRK theory. Items 1-6 are supported by Kim et al. (2013). 7-9 are new predictions from incorporating new memory types.

	Prior Predictions
(1)	Learning follows the power law curve of learning Time = A + BN^{-C} (A, B, C are constants)
(2)	Three stages of knowledge: a. Acquiring declarative and procedural knowledge b. Consolidating the acquired knowledge c. Tuning the knowledge towards overlearning
(3)	Retention of declarative knowledge decays quickly and catastrophically
(4)	Retention of mixed declarative and procedural knowledge decays moderately
(5)	Retention of proceduralized knowledge has least decay
(6)	Recognition and perceptual-motor knowledge have different learning curves than procedural or declarative.
	New Predictions
(7)	Ideal training schedules will vary by knowledge-type; perceptual-motor may require minimum training block size
(8)	Retention of perceptual-motor knowledge appears to decay little
(9)	Recognition memory is (probably) not fragile

Method

We will first describe a complex task that we use and for which we have some data and a running model that learns. We will then describe the architecture and the model, which serves as a subject in this simulated study. We then describe the human data, and how we ran the model.

The Dismal task

The Dismal task, illustrated in Figure 2, is a spreadsheet task that can be used to measure procedural knowledge and skills learning and decay (Kim & Ritter, 2015). The Dismal task was created to be done in the Dismal spreadsheet (Ritter & Wood, 2005). Dismal is an open source, extendable spreadsheet in Emacs.

The overall task length and subtask variety make Dismal a relatively complex task. There are 14 different subtasks in Dismal (Table 2). The subtasks contained attention shifts, encoding of information, attending to information, key presses, and mouse moves/clicks. Previous work using Dismal allows comparison to human data and model predictions.

These tasks can be done with two different interfaces—(a) a keyboard with key-based commands, or (b) a mouse or vertical mouse. The vertical mouse provides new motor skills to learn and forget because it requires a different hand posture.

Table 2. The 14 Dismal spreadsheet subtasks.

	Dismal task sequence
(1)	Open a file, named normalization.dis under the "experiment' folder
(2)	Save as the file with your initials
(3)	Calculate and fill in the frequency column (B6 to B10)
(4)	Calculate the total frequency in B13
(5)	Calculate and fill in the normalization column (C1 to C5)
(6)	Calculate the total normalization in C13
(7)	Calculate the length column (D1 to D10)
(8)	Calculate the total of the "Length" column in D13
(9)	Calculate the Typed Characters column (E1 to E10)
(10)	Calculate the total of the "Typed Characters" column in E13
(11)	Insert two rows at A0 cell
(12)	Type in your name in A0
(13)	Fill in the current date in A1 using the command
(14)	Save your work as a printable format

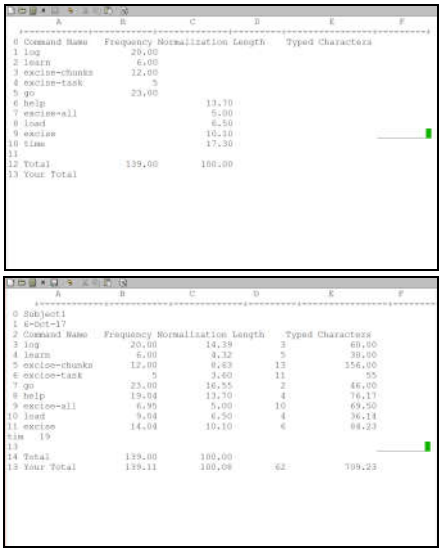


Figure 2. Dismal interface with initial state (top) and final state of the task (bottom).

The two interface modes can be used to study different types of knowledge: recall of keystroke commands and recognition of menu-based commands.

ACT-R

ACT-R is a theory of the mechanisms that make up cognition. It is an example of a unified theory of cognition (Byrne, 2012; Newell, 1990), in that intends to predict and explain human behaviour by simulating the steps of cognition with a fixed set of mechanisms. ACT-R predicts behaviour and activation of brain regions by using mechanisms including procedural and declarative knowledge, and working memory as activation, to perform tasks.

We briefly review ACT-R's components and then the memory equations; other more complete treatments are available (e.g., Anderson 2007; Anderson 1982). Thus, we briefly review ACT-R's components and then the memory equations.

The architecture components

ACT-R consists of modules and buffers. Modules are responsible for processing one kind of information and are the mechanisms for modifying and implementing a buffer; buffers are contents that are visible to other modules. The modules descriptions and roles include:

the Visual module is for identifying objects in the visual field. Visual objects and their identities are located in the visual buffer and visual location buffer and monitoring attention and visual objects such as scanning a computer screen.

the Manual module is for controlling the hands; the manual/motor buffer handles controlling and monitoring hand movement such as typing on a keyboard.

the Declarative module is for retrieving information from memory, and a Goal module is for keeping track of current goals and intentions. The Goal buffer stores the current subgoal step and its next step.

A central production system is a rule-based system that performs the matching, selection, and execution of production rules. Also, it coordinates the communication and performance of these modules through the application of production rules. The central production system works in parallel with modules and constantly updates and queries the buffers' data.

The memory equations

Throughout the task completion by an ACT-R model, each declarative memory used will have its base-level activation increased. Presentation of an item (or chunk), and thus subsequent changes in base-level activation, can occur at three points in the process: at item/chunk creation, at the time when two items are merged, and when the chunk itself is retrieved.

This process has two mutually exclusive options for how to calculate learning for chunks during the task procedure: the Optimized Learning Equation (OL), and the Base-Level Learning Equation (BL),. These are shown in Equations 1 and 2. Parameters (e.g., :bll) refer to specific values set within ACT-R's base configuration.

Equation 1: The Optimized Learning Equation (OL)

$$B_i = \ln\left(\frac{n}{1-d}\right) - d * \ln(L) + \beta_i$$

***n*:** The number of presentations for chunk *i*
***d*:** The decay parameter set using the :bll parameter
***β_i*:** A constant offset set using the :blc parameter

Equation 2: The Base-Level Learning Equation (BL)

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d} + \beta_i\right)$$

***L*:** The lifetime of chunk *i* (the time since its creation).
***t_j*:** The time since the *jth* presentation. A presentation, or reference, is either the chunk's initial entry into DM or when another chunk is merged with a chunk
Unlabeled variables in BL are shown in Equation 1.

The two equations differ in their accuracy and computational cost. BL is costlier because it accounts for time through the t_j parameter (with associated repeated exponential computations based on t_j's), while OL simplifies the equation to primarily rely on the number of presentations.

The model

Herbal, a high-level behaviour representation language, creates ACT-R source code (Ritter et al., 2006). Herbal has been used to build several ACT-R models of this spreadsheet task that ranged in expertise from novice to expert. Further details about the mode can be found in Paik, Kim, Ritter, & Reitter (2015). We used the novice model with 9 initial rules and 520 declarative memory elements because it starts the task using declarative knowledge.

Number of runs

Completion time curves were generated from the mean time for a given data point based on multiple runs of the model for the task (N=5). This number is sufficient because we are observing broad trends rather than focusing on specific effect sizes (Ritter, Schoelles, Quigley, & Klein, 2011).

Existing human data

Participants in Kim's (2008; Kim & Ritter, 2015) study (N=60) were divided equally into two groups: one used only the keyboard (N=30), and another group used the combination of vertical mouse and keyboard (N=30). They completed the Dismal task and came back at 6, 12, or 18-day intervals. We used the vertical mouse interface subjects' data. The human data for days 1-4 matches the model predictions for trials 1-4 shown in Figure 3.

Results

We ran a series of models to explore how the ACT-R model of the Dismal task predicts learning and decay. For each test, the model was run with the OL and BL equations to compare their predictions. We found similarly shaped retention curves for both, but there were differences as well.

Predictions with Optimized Learning and Decay

Figure 3 shows a standard learning curve on task time where the model was run over 10 trials without delays between trials. Decay curves are shown for up to 5 days decay.

This is how most repeated trial ACT-R models are run. They are run multiple times, with no time between trials in the model, even if there is time between human trials for the subjects, which there was for this data set–each trial was run on a separate day.

These results are consistent with most of the predictions in Table 1. (1) Task completion times on the solid black line followed the shape predicted by the power law of learning. (2) The three stages are there, but when examining the model trace, we see that proceduralization starts in the first

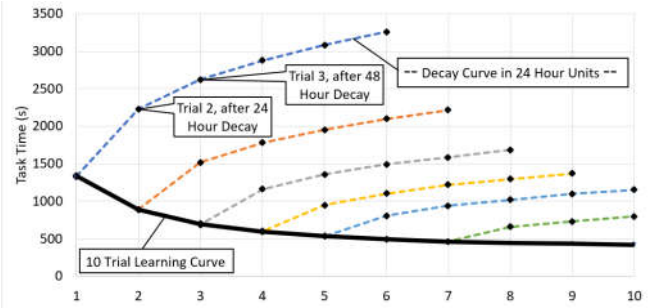


Figure 3: Predictions for task time with OL (black). Forgetting curves show task time after [1-5] days of decay after a period of consistent practice.

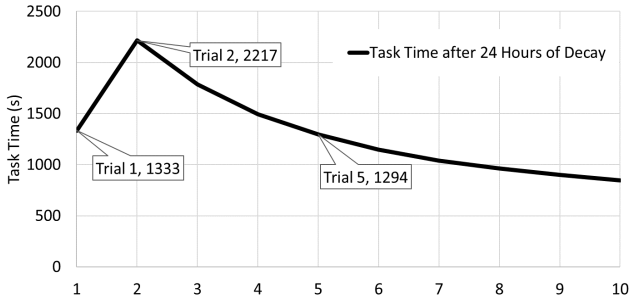


Figure 4: Predictions for task completion time with OL for 10 trials with 24-hour decay periods between each trial.

task of the first trial. (3) Declarative knowledge decays but did not do so catastrophically for this model and task. (4 & 5) the decay of the mixed knowledge (middle trials) was slower than the declarative knowledge but not as slow as the procedural knowledge (later trials). We could not examine the other predictions (7,8,9) with this task.

We found unexpected results from applying a decay period in the model. After one practice and 24 hours of decay, the task time was slower than during the first trial. Decay caused a worse time than a novice's first trial. The mean task time for trial 1 was 1335 s. After the 24-hour decay period, task time for trial 2 was 2228 s compared to 882 s for the normal curve. In short, decay causes memories (and thus performance) to be worse than the very first trial of the experiment.

So, we explored the effects of decay by including a 24-hour decay period after each practice to simulate what the subjects did, that is, practice and then wait a day until the next practice. These results are shown in Figure 4.

Figure 4 shows a typical learning curve after trial 2. The initial 24-hour decay causes performance to be longer on day 2 than on the first day. The performance does not get as fast as day 1 until trial 5. Including the time between practices does not improve our predictions. We do not include forgetting curves because the learning curve is unusual.

Predictions with base-level learning equation

Next, we used the base-level learning equation for the same process. Figure 5 shows the decay and retention curves through 10 trials of practice with the base-level learning

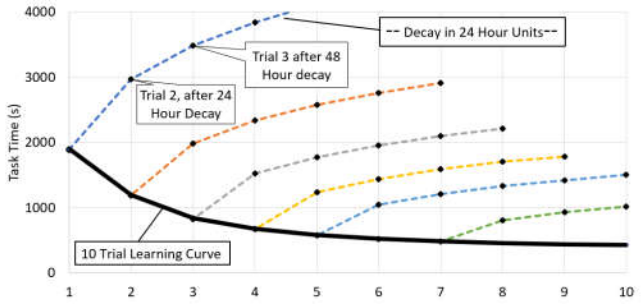


Figure 5: Predictions for task time with BL for ten trials of practice (black). Forgetting curves show task time after [1-5] days of decay after a period of consistent practice.

Figure 6: Predictions for task completion time with BL for 10 trials with 24-hour decay periods between each trial.

equation instead of the optimized learning equation. Figure 5 is without time between trials and Figure 6 is with 24 hours between trials. In Figure 6 we do not include forgetting curves to emphasize the unusual shape of the learning curve.

Overall, the learning and retention curves with BL are shaped similarly to the OL model predictions but start with a higher mean task time at trial 1 (1901 s). The final task time was 424 s, or 22% of the first trial. Again, this showed that the performance on trial 2 following 24-hour decay was worse than the initial time.

Predictions with initial day delay and day delays inserted

We then considered applying a day delay in the initial declarative knowledge before starting to perform the task. This could represent less complete declarative learning of the task knowledge. So, we set the model to run with a day delay after the declarative knowledge has been first learned.

Figure 7 (top) shows three learning curves using the OL equation, and the BL equation (bottom). The red dashed curve is the learning curve without time decays; it is the same as Figure 3. The blue dotted line is with 24 hours between each trial, the same as Figure 4. The black solid line has a 24-hour decay before the first trial and after each trial.

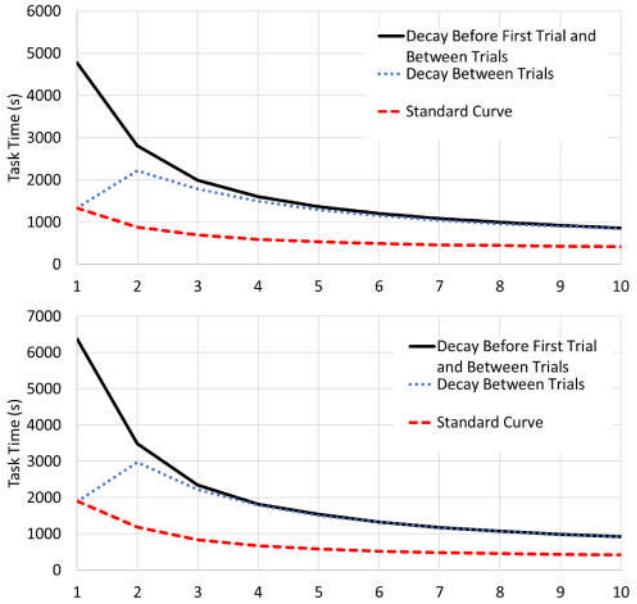


Figure 7: Predictions for task completion time with OL (top) and BL (bottom) with 24-hour decay periods between each trial and an initial 24 decay on the declarative knowledge.

Final model, base-level learning with day delays and retention curves

The predictions in Figure 7 remain somewhat unsatisfactory. They either leave out the decay between practice trials (but fit the data fairly well), or they include the decay, but over predict the task time. We thus tried an adjustment to the decay parameter suggested by Lebiere (personal communication, October 2017).

This adjustment provides different decay constants for within and without an experimental situation. This change probably represents the effect of proactive interference more accurately in that during a study the task-related memories are more similar in a time block in a study than they are outside a study. With non-study time, the decay time is reduced to be ¼ of the actual delay time, that is, instead of 24 h between trials, only 6 hours is added as decay after learning the task knowledge and between trials. Figure 8 shows this model's predictions.

Discussion and Conclusions

This model provides several insights about learning and the ACT-R memory equations.

Insight: There are no fixed learning stages

The results of the model's learning show that there are no crisp divisions between the learning stages, contrary to predictions made in previous theories for the learning curve – there are no points of inflection in the learning curve to show the transition. In the knowledge used in the model (declarative and procedural), there are also not inflection points. The model thus predicts that there are not distinct stages in this task at least, where the knowledge is

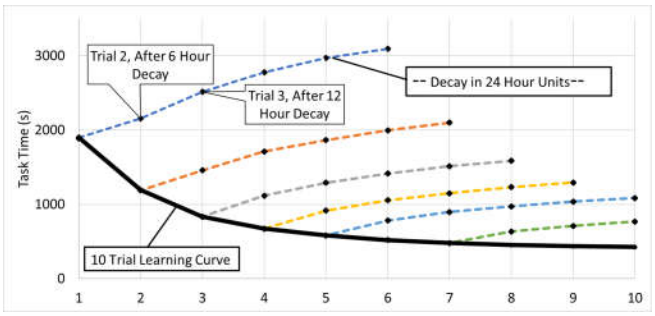


Figure 8. Predictions for task completion time with BL with 24-hour decay periods between each trial and an initial 24 decay on the declarative knowledge, adjusted

all declarative or all procedural. Subtasks are more clearly one or the other because only one type of knowledge might be used in a smaller task. However, the model shows that even during the first trial, knowledge is being not only proceduralized, but also that the procedural knowledge is being used. So, partway through the first trial, the model has mixed task knowledge in some technical sense.

We suggest that the stages be relabeled into mostly declarative, mixed, and nearly all procedural. This is in contrast to [all] declarative, mixed, and [all] procedural.

Insight: the stages will vary by task and strategy

These stages will also depend on the task components and the task distribution. If the task is primarily a declarative task, it will basically stay a declarative task. If it is a perceptual-motor task or a procedural task, it will transfer into a proceduralized task.

This model also shows that location of these stages will vary by task. Tasks with high declarative components will remain in the mostly declarative stage longer, according to this definition, because more parts that will stay declarative.

If an unusual task from the distribution of possible tasks comes along, the learner may be shifted back towards more declarative task knowledge, or as Rasmussen (1983) notes, knowledge-based control knowledge. So, distribution of tasks that all occur equally will have different learning and stages than a distribution of same tasks (and knowledge) where some only occur rarely.

Insight: implications for empirical studies

The model provides some implications for running a human study in this area. To measure the decay curve, you must have at least three decay points or two points and a strong theory. A study with humans cannot reset the model to get multiple decay measurements but will have to train a subject and then can only measure decay once without retraining occurring. To measure these points, you must train subjects to standard, and then have them come back at a delay. You can only have them come back once, because the measure is a training. Thus, to measure the decay at three points, you have to each data point be its own condition. To study decay after training 1, 2, and 5 days, and decay at 4, 8, and 16 days, you require nine groups. Thus, the decay graph 9x

more expensive (assuming subject drop out does not increase because of the delay) than the simple learning curve. This is partly why these curves are studied less.

Insight: The decay curve at one day is practice on the next day

Consider that you will be training every day and wish to study the decay function after 1, 2, and 5 consecutive practice days. If you measure the amount of decay after one day, the performance after one day of decay when you are training once per day is the same as the group that is on the training schedule. Thus, the decay curve in such circumstance has a decrease in performance time with one day decay (if the test includes training such as doing the task), and then decreased performance on later days. This may be like walking as being way of falling forward.

Insight: The subtask curves within most instruction are a mix and have varied retention intervals

Very few real-world tasks will have this pure of a training and retention schedule. In the real world, after an hour of training, the learner will move into new material in later sessions. Thus, the learning and retention curves will include multiple small curves, and some subtasks will be trained every session and get much faster, and some tasks will occur only rarely and will have long decay times (if learned early) or short decay times (if learned later). You would need a computer to keep track of them!

Future Work: How to run a study of a complex task to test the KRK theory

Based on these results, in our empirical test of the KRK theory we will examine a complex task, a trouble shooting task, with 1, 2, and 5 practice trials, separated by a day per trial. We will look at performance at 3, 5, and 7 days decay after the last practice. We will not have the full training material at the decay tests, just the trouble shooting tests. This will measure the decayed knowledge with little or no relearning of it.

Acknowledgments

Kevin Gluck gave useful comments about this model in a discussion at ICCM. This project was supported by ONR grant N00014-15-1-2275.

References

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

Byrne, M. D. (2012). Unified theories of cognition. *Wiley Interdisciplinary Reviews: Cognitive Science, 3*(4), 431-438.

Destefano, M. (2011). *The mechanics of multitasking: The choreography of perception, action, and cognition over 7.05 orders of magnitude.* Unpublished PhD thesis, Cognitive Science, RPI.

Fitts, P. M. (1964). Perceptual-motor skill learning. In A. W. Melton (Ed.), *Categories of human learning* (Vol. 47, pp. 381-391). New York: Academic Press.

Kim, J. W. (2008). *Procedural skills: From learning to forgetting.* Unpublished PhD thesis, Department of Industrial and Manufacturing Engineering, The Pennsylvania State University, University Park, PA.

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.

Kim, J. W., Ritter, F. E., & Koubek, R. J. (2013). An integrated theory for improved skill acquisition and retention in the three stages of learning. *Theoretical Issues in Ergonomics Science, 14*(1), 22-37.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Paik, J., Kim, J. W., Ritter, F. E., & Reitter, D. (2015). Predicting user performance and learning in human-computer interaction with the Herbal compiler. *ACM Transactions on Computer-Human Interaction, 22*(5), Article No.: 25.

Posner, M. I. (1973). *Cognition: An introduction*. Glenview, IL: Scott Foresman.

Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13*, 257-266.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering.* New York, NY: Elsevier.

Ritter, F. E., Haynes, S. R., Cohen, M. A., Howes, A., John, B., Best, B., et al. (2006). High-level behavior representation languages revisited. In *Proceedings of ICCM - 2006- Seventh International Conference on Cognitive Modeling*, 404-407. Edizioni Goliardiche: Trieste, Italy.

Ritter, F. E., Schoelles, M. J., Quigley, K. S., & Klein, L. C. (2011). Determining the number of model runs: Treating cognitive models as theories by not sampling their behavior. In L. Rothrock & S. Narayanan (Eds.), *Human-in-the-loop simulations: Methods and practice* (pp. 97-116). London: Springer-Verlag.

Ritter, F. E., & Wood, A. B. (2005). Dismal: A spreadsheet for sequential data analysis and HCI experimentation. *Behavior Research Methods, 37*(1), 71-81.

Seibel, R. (1963). Discrimination reaction time for a 1,023-alternative task. *Journal of Experimental Psychology, 66*(3), 215-226.

Tehrachi, F., Oury, J.D., Ritter, F. E., Predicting learning and retention (forthcoming).

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

VanLehn, K. (1996). Cognitive skill acquisition. *Annual Review of Psychology, 47*(513–539.

An SGOMS Model of Human StarCraft Game Playing in Autonomous Agents

Chad A. Peters (chad.peters@carleton.ca),
Robert L. West (robert_west@carleton.ca),
Babak Esfandiari (babak@sce.carleton.ca)
Institute of Cognitive Science, Carleton University, Ottawa, Ontario, Canada

Abstract

This paper uses cognitive modeling to investigate how human players are still able to beat artificial intelligence (AI) players in tournament games of StarCraft. To do this we modeled human game play using the SGOMS macro cognitive architecture. Using the same perceptual motor functions and the same strategies and rules as the AI player, the SGOMS player produced significantly more wins. Possible reasons for this are discussed.

Keywords: GOMS, SGOMS, StarCraft, Macro Cognition, Dynamic Systems.

Introduction

As recently demonstrated by the victory of AlphaGo (Silver et al., 2016) over a high ranked GO master, artificial intelligence for games is reaching a level that exceeds human abilities. However, there is still at least one game that humans dominate; a Real-Time Strategy (RTS) game called StarCraft[1]. Despite the best efforts of researchers in the field of Artificial Intelligence, autonomous systems have yet to claim the title of World Champion against the best professional StarCraft tournament players (Kim & Lee, 2017). The game is relatively simple in design, and yet requires a comprehensive skillset including resource management, decision making under uncertainty, opponent modeling, and real-time planning; all of which no one autonomous system has been able to produce at the level of a professional human player (Ontanon et al., 2013).

To better understand how humans win in StarCraft we played a baseline StarCraft agent against the same StarCraft agent modified to use the SGOMS architecture. SGOMS (West & Nagy, 2007) is a theory of human expertise and is well suited to model StarCraft as the StarCraft play relies heavily on executing fixed memorized strategies in a dynamically changing environment, which is what SGOMS is designed to do (West & Nagy, 2007). The reasoning was as follows; if SGOMS is a valid model of human expertise then it might capture the ability that allows humans to win in StarCraft.

To test this theory, we played an SGOMS agent against the baseline (herein "standard") StarCraft agent in a tournament format to record and capture a variety of metrics such as win ratio over time, resource utilization, and action rates. Knowledge, actions, and perception were held constant across the models; that is, both models had access to the same information from the game, could execute the same game commands, and had the same memorized strategies. The only difference was the way the expertise was processed.

The standard agent was programmed using best practices for this type of game, including transition points between strategic moves, as well as the use of encapsulating routine interactions in procedural functions. The proper transition between game states is important to ensure a constant collection and flow of resources to the proper units, or unit-producing buildings, such as to maximize the chances of survival regardless of the passage of time. Examples of states in a game of StarCraft include maximizing the number of resource collection units for an area, expanding the base when necessary, creating additional supply depots before the maximum unit count is reached, attacking the enemy when possible, and retreating when appropriate to counter-attack at a later time.

Of equal importance to a game-playing agent is how the underlying logic is represented and implemented in code. For example, many routine actions will be used regardless of game state (building, attacking, defending), and can be abstracted into proper functions, such as collecting environmental information, or selecting and moving units. In the case of the standard agent, all environmental factors and possible moves were considered for every frame, and like many agents with comparable architectures, produced hundreds of actions *per second*, which in most situations can be handled by the client without slowing down in either simulated- or real-time. This kind of behavior is very unlike the way a human being plays, as most world-class professional StarCraft players will produce around 300 to 400 Actions Per Minute (APM) at peak intervals (Lewis, Trinh, & Kirsh, 2011).

In contrast to the standard agent, the SGOMS model operated according to the SGOMS architecture, which caused it to run considerably slower due its more centralized and deliberate processing design. Therefore, in terms of processing speed, the standard model had an advantage. Assuming all other things to be equal, the standard model should have been the better player, but this is not what we found.

The Macro-Architecture Hypothesis

SGOMS is a macro cognitive architecture (West & Macdougall, 2014). The macro architecture hypothesis states that Newell's (1990) cognitive band (see Figure 1), can be divided into a *micro* cognitive level and a *macro* cognitive level, where the macro cognitive level describes how we execute and manage higher-level functionality such

[1] https://starcraft.com

as expertise, problem solving, strategic thinking, etc. (see Figure 2). SGOMS has been implemented using the ACT-R micro cognitive architecture (see West & Pronovost, 2009), but in this case, SGOMS was implemented using the underlying code for the StarCraft agents.

Scale (sec)	Time Units	System	World (theory)
10^{7}	months		
10^{6}	weeks		Social Band
10^{5}	days		
10^{4}	hours	Task	
10^{3}	10 min	Task	Rational Band
10^{2}	minutes	Task	
10^{1}	10 sec	Unit task	
10^{0}	1 sec	Operations	Cognitive Band
10^{-1}	100 msec	Deliberate act	
10^{-2}	10 msec	Neural circuit	
10^{-3}	1 msec	Neuron	Biological Band
10^{-4}	100μs	Organelle	

Figure 1: Newell's System Levels

The SGOMS Framework

SGOMS, or Sociotechnical GOMS, is a type of GOMS model (Card et al., 2008). The unique feature of SGOMS is that it adds a structure above unit tasks called *planning units*. Planning units are larger flexible units that can be interrupted, switched, and restarted where the agent left off.

The purpose of planning units is to allow agents to quickly adapt to unexpected interruptions. Planning unit choice is based on constraint-based decision-making and is strategic in nature. Constraints are continuously updated through parallel perception functions and interruptions are triggered by a parallel monitoring function (West & Nagy, 2007).

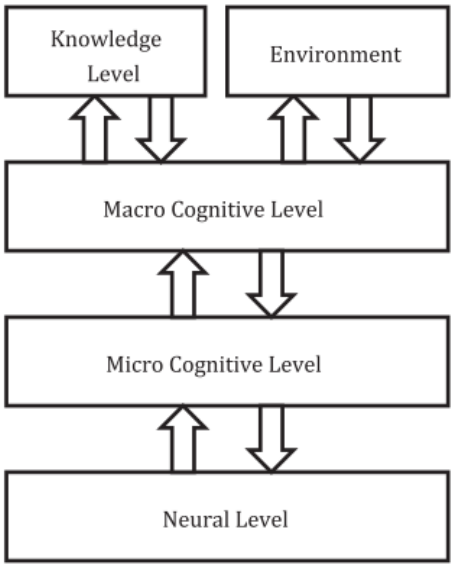


Figure 2: The Macro Cognitive Hypothesis

Real-Time Strategy Environments

Real-Time Strategy (RTS) games provide a challenging landscape for gamer and scientist alike. RTS games possess a number of distinct differences over alternative forms of competitive strategy games (such as Chess or Go), and force players to rely on partial information while pursuing multiple complex goals; and usually all at the same time (Weber & Mateas, 2008).

The StarCraft game presents the user with a selection of three different alien races, each with their own unique blend of strengths and weaknesses that have been balanced by the game designers using nearly two decades of online competitive play. The StarCraft game universe requires players to develop and utilize a number of specialized skillsets that are, at the time of this writing, a considerable challenge for expert agents; these challenges can be organized into recurring organizational themes (Ontanon et al., 2013) such as Resource Management, Decision making under uncertainty, Spatial and Temporal reasoning, Collaboration between multiple humans/agents, Opponent modeling and learning, and Adversarial real-time planning.

Ontanon et al. (2013) also make an interesting distinction between how these logical organizations manifest as major architectural units in a large sample of competitive bots, and how the StarCraft community divides their own human behavior into two types of tasks; *micro* and *macro*. This distinction is consistent with the distinction between micro and macro architectures proposed in the Macro Architecture hypothesis (West & Macdougall, 2014).

Micro behavior in most RTS games is characterized by fast, reactive tactical maneuvering of small preconfigured groups of units. This would be akin to the squads found in standard military parlance, and in RTS gameplay requires a sharp focus and quick reflexes to survive the hit-and-run techniques so prevalent in competitive play.

Macro behavior, in comparison, is all about the long timeframe and strategic choices in base building, expansion, and choosing the best time for investing in technology upgrades that benefit all units instead of simply building more units; these are all character behaviors ascribed to the seasoned battlefield commander who knows when to initiate a corrective course and switch to a more appropriate strategic plan.

The Standard Agent

The first StarCraft agent implemented a set of basic procedural rules in order to provide a *baseline* of rudimentary behavior capable of finishing a standard game. Action classes included the operations required to build a base, create and replace units, and attack the enemy. We added strategic logic in order to manage resources, find and attack the nearest enemy units or buildings, and retreat when appropriate.

These conditionals are, in effect, an oversimplification of the code required to interface with the game but will serve for the purpose of this example. Using these principles, we created a simple reactive agent to walk through the

following constraint-based decisions until the game context matched the most appropriate production rule:

- If a Barracks is ready, train another Marine.
- If we are Supply Blocked, build a new Depot.
- If we have resources, build a new Barracks.
- If we have at least 10 Marines, attack the Enemy.
- If we see the Enemy, and have less than 10 Marines, retreat to our Command Centre.

The entire series of production rules was invoked once every in-game frame (StarCraft runs at 24 frames per second), with every matching rule triggering the associated operator action.

StarCraft Brood Wars API

Our StarCraft agent was implemented using a proxy layer that allows a third-party developer to directly interact with the game in real-time, without manual control of any system peripherals. The Brood Wars Application Programming Interface (BWAPI)[2] was designed for this purpose; it provides a standard definition for agents written in C++ to read the game state, make intelligent decisions, and send commands to the game in a fashion similar to a human player.

The agent was originally written using the BWAPI tutorials designed to produce a game-playing agent capable of progressing through all game states required to reach the final state of either winning or losing by total annihilation. In this case, the agent is able to manipulate units, build buildings and units, form groups of units, move across the map, and attack opposing forces. Every command that is available to a human player can be interpreted as GOMS operator using BWAPI, with the exception of panning the screen using the mouse cursor.

The SGOMS Agent

The next iteration of the standard agent was based on the principles of SGOMS theory. In order to test the viability and impact of this architecture, all standard agent logic was, in principle, left alone. This approach simplified the application of SGOMS theory, freeing us to classify each conditional block into a hierarchy of operators, methods, unit tasks, or macro-level planning units (see Figure 3), all depending on estimated system-level action times.

Operators

Operators are the lowest level of system interaction and can be completed in milliseconds. In terms of the motor actions used by humans in the StarCraft game, any immediate operations involving a single mouse click or depressing a keyboard button was identified as a system level operation.

Example operations would include selecting a building, creating a new unit at a pre-selected building, selecting a new unit, or issuing a command to a pre-selected unit. These operators were abstracted in order to provide a general function interface for more complex grouping of operations.

Methods

Methods are a sequence of operators performed as a group to accomplish an efficient task without interruption. These sequences take from 1 to 3 seconds and require the focused attention of an expert player; a momentary diversion from the "main action" of what is going on in the larger context.

Example methods would include grouping units, or creating a building at a specified location on the map. Most of the methods used two or three Operators fired in a ballistic fashion without opportunity for interruption through arousal.

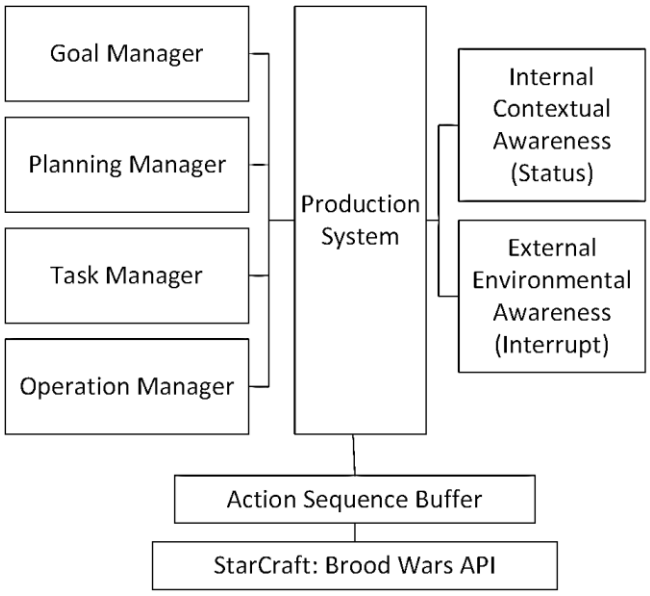


Figure 3: SGOMS Agent Architecture

Unit Tasks

Methods and operators are performed in groups to accomplish unit tasks. Unit tasks in SGOMS are strategically controlled by planning units. A unit task can fire independently, or as part of a chain of tasks performed in quick succession, or as one of a few options selected in a moment based on constraint-based decision making. According to Newell's (1990) system levels, a typical unit task would require between 3 and 10 seconds for an expert player to accomplish.

Example unit tasks would include grouping units into a squad, queueing rally and patrol points, or building a series of buildings or units in succession.

[2] https://bwapi.github.io/

Planning Units

Strategic decision making is found in the higher system levels of Newell's cognitive band and is the main contribution of the Macro layer of SGOMS theory. As described by (West, 2013), there are three main classes of planning units - situated, ordered, and identity. *Situated* planning units will queue and execute a unit task if the constraints of the environment are satisfied. *Ordered* planning will perform a pre-defined series of unit tasks such as used in a first-in-first-out (FIFO) queue. *Identity* planning units are the simplest form and will execute a specific unit task to accomplish the goal.

Context and Situational Awareness

The Internal Contextual and Situational Awareness modules represent how the player interprets the game and what they know about the current state of the game. This module uses situational information from Intelligence gathering and feeds the top-level executive control of the player agent.

Similar to the standard agent, the Contextual factors used by the SGOMS agent's executive control included tracking primary resources such as mineral and unit supplies (see Figure 4 overlay), transitions between defensive and offensive states of combat, and tracking the nearest known enemy position.

Figure 4: SGOMS Agent Playing StarCraft

A Real-Time Strategy Game defines a series of overarching goals that must be accomplished (sometimes in no specific order), to bring the player closer to a winning state. For example, in an adversarial context with multiple squad-level and unit-level tactics, a series of offensive and defensive maneuvers, coupled with gathering intelligence, planning base expansions, territory control, and resource management, must all be performed in order and unison. In some cases, goals must be paused or abandoned if critical interruptions occur, such as enemy units attacking the player's critical units or structures.

Planning units in StarCraft can be broadly classified as defensive or offensive in nature, depending on enemy force involvement. Defensive maneuvers would include scouting raw materials, repairing a base, building or replenishing a group of units, or responding to an attack. Offensive planning units would include scouting to find enemy positions, and directly attacking an enemy force.

External Monitoring and Interrupt System

The External Monitor module serves as an interface module between procedural/motor sub-systems (Operator interface) and receptors as reported by the Brood Wars API. This system acts as the buffer between audio/visual game cues and whether to ignore the cue or escalate to Situational Awareness for Planning Unit interruption. This module acts at the higher Biological band, as an expert player will require around 100ms to 300ms to react to arousal cues; in this case, a reaction could be choosing to ignore a distraction based on prior experience.

The SGOMS agent handles a variety of game-supplied stimulus types including text messages, discovering a new unit, destruction of a unit, and supply shortages; all of these events would be interpreted by a game player as

unmistakable auditory or visual cues, and used to signal the interruption of the existing Planning Unit when appropriate.

Experimental Design

Preliminary tests of both the Standard and S-GOMS agents confirmed that the simple production logic used in both was not enough to win against the default A.I. provided with the game. This came as no surprise to the authors, as the original StarCraft agents will quickly adopt newer units and buildings and win through technical superiority against most early-game strategies. This begs the question; if a Standard agent cannot win against the default A.I., how then can we viably measure the effects of an equally bad (or possibly worse) variant of itself?

Considering the purpose of this research is to measure the effects of using a human-inspired cognitive architecture, we deliberately chose to compare the Standard agent against the SGOMS variant of itself, with no more than a refactoring of the same GOMS-inspired production rules into a Macro-Cognitive implementation. This was done such that all else being *truly* equal, we can measure the effects of an SGOMS implementation; an approach that could be extended to any such agent.

We compared the Standard and SGOMS agents in a 1 vs 1 tournament configuration comprised of 10 maps used at previous competitions run at SSCAIT (Čertický & Churchill, 2017), with 3 rounds per map, for a total of 30 games. The tournament maps cover a spectrum of terrain styles and sizes, and were originally designed to cater to matches involving between 2 and 8 players total. The test environment was composed of three virtual machines running on Windows 10, each having an identical configuration apart from tournament-specific roles. One of the machines served as the tournament server, while the other two were registered clients, all contained within the same local area network.

The BWAPI Tournament Manager was configured to alternate agents between test machines to eliminate any machine-specific advantage. We also selected the standard range of map sizes with randomized starting locations to account for location advantage and the effects of travel distance between bases. Once all 30 rounds were concluded, the win/loss results as well as replay files were saved for comparison and insights for further in-game analysis.

Results

As shown in Figure 5, the SGOMS model won, by far, the most games. The replay files were post-processed using BWChart[3] for visual inspection and to derive metrics for each agent during the match. Of primary note was the effect of combat intensity on agent APM, and the difference in APM during observable skirmishes.

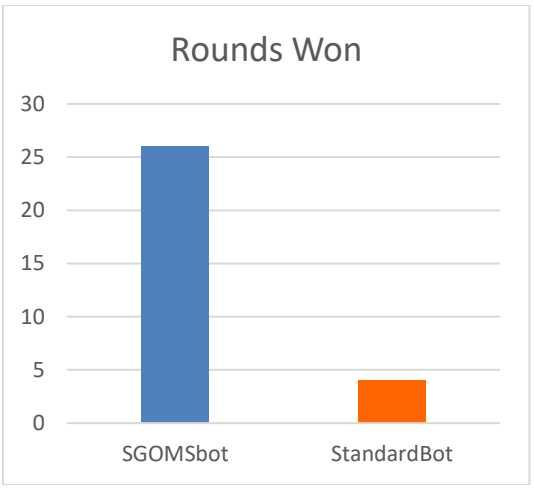


Figure 5: Tournament Rounds Won per Agent

Figure 6 shows APM in a typical game. The standard agent (red) has a higher APM for most of the skirmishes, and only after it flatlines (combat units destroyed) does the SGOMS agent APM (blue) go up as it cleans up the leftover worker units and buildings. Further visual inspections revealed a similar pattern across matches.

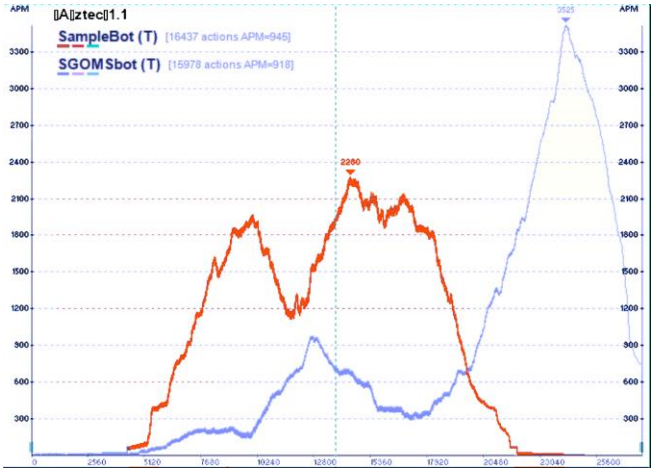


Figure 6: Actions-per-Minute across a replay

From observations we also noted that the standard agent sometimes produced a style of play that looked more "jittery" than normal human play; for example, if the game state reached an indeterminant position, units movements would quickly oscillate between competing priorities. This observation could be caused by the ability to more quickly adjust the behaviors of individual units in the game.

Conclusions

The standard and SGOMS agents were identical in terms of the information they had access to and the rules they used to create responses. And yet, the SGOMS agent was clearly the

[3] http://liquipedia.net/starcraft/BWChart

better player. This occurred despite the fact that SGOMS was significantly slower, which should produce a disadvantage in this game. Another interesting finding was that SGOMS produced an APM rate that was often similar to the human APM for this game, roughly 300-600 APM (Lewis, Trinh, & Kirsh, 2011). This may be coincidence, but it raises the possibility that the human APM is also due to cognitive processing, rather than perceptual/motor speed alone.

The standard agent was clearly more active (faster) than SGOMS, yet it lost the majority of the games. However, as West and Lebiere (2001; see also West et al., 2005) demonstrated, when the outputs of two competing agents feed back on each other it can create a coupled system with hard to predict dynamic properties. Also, unlike chess or GO, StarCraft has been tuned to human play. The number of units, the terrain, and the construction times are all tuned to make it a challenging but playable game for *humans*. In other words, the dynamics of the game may be tuned to respond to specifically human cognitive abilities. More research is required to further understand these results.

In terms of SGOMS, the results were encouraging. Not only did the SGOMS agent show a closer resemblance to the result of humans being able to beat a faster and seemingly more powerful opponent, it also produced an APM rate closer to that of human players. The results also suggest that the dynamics of certain types of video games may have evolved, through feedback, to resonate with *human cognitive abilities*. Unlike chess or GO, to win this type of game artificial agents would need to be more human-like, not faster or more powerful. In this regard, the architectural distinction between micro and macro cognition may be of critical importance.

Acknowledgments

This research was supported in part by the Carleton Cognitive Modeling (CCM) Lab, and the Network Management & Artificial Intelligence Lab at Carleton University, Ottawa.

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*.

Kim, Y., & Lee, M. (2017, November). Humans Are Still Better Than AI at StarCraft—for Now. Retrieved from https://www.technologyreview.com

Lewis, J., Trinh, P., & Kirsh, D. (2011). A corpus analysis of strategy video game play in starcraft: Brood war. *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, 687–692.

MacDougall, W. K., West, R. L., & Hancock, E. (2014). Modeling Multi-Agent Chaos: Killing Aliens and Managing Difficult People. *36th Annual Meeting of the Cognitive Science Society*, 2603–2608. Retrieved from https://mindmodeling.org

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, Massachusetts: Harvard University Press.

Ontanon, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., & Preuss, M. (2013). A survey of real-time strategy game AI research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in Games*.

Pronovost, S., & West, R. L. (2008). Bridging Cognitive Modeling and Model-Based Evaluation: Extending GOMS to Model Virtual Sociotechnical Systems and Strategic Activities. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *52*(19), 1635–1639.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., … Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*.

Weber, B. G., & Mateas, M. (2008). Building Human-Level AI for Real-Time Strategy Games.

West, R. L. (2013). The Macro Architecture Hypothesis: A Theoretical Framework for Integrated Cognition. *AAAI Fall Symposium Series*, 102–108. Retrieved from https://www.aaai.org

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

West, R. L., & Macdougall, K. (2014). The macro-architecture hypothesis: Modifying Newell's system levels to include macro-cognition. *Biologically Inspired Cognitive Architectures*, *8*, 138–147.

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. L., & Pronovost, S. (2009). Modeling SGOMS in ACT-R:Linking Macro- and Microcognition. *Journal of Cognitive Engineering and Decision Making*, *3*(2), 194–207.

Modelling metareasoning about decision thresholds in a perceptual learning task

Vasundhara Rakesh
Department of Chemical Engineering
IIT Kanpur

Nisheeth Srivastava
Department of Computer Science and Engineering
IIT Kanpur

Abstract

The mathematical relationship between the drift diffusion model and sequential probability ratio testing implies that observers choose thresholds to optimize some desired level of accuracy across the entire set of trials. We argue that it is more cognitively natural to assume that decision-makers set this threshold adaptively, using information from recent trials to adjust it for upcoming ones. To test this hypothesis, we designed and conducted a random dot motion discrimination experiment where the coherence parameter that controls task difficulty varies across trials in a predictable manner. We found that observers are quicker to respond in trials that follow successively easier trials and vice versa. We designed a hierarchical drift diffusion model that allows decision-makers to adapt their evidence threshold based on the trend of difficulty of previous trials and show that it fits our experimental data better than a simple drift diffusion model.

Keywords: drift diffusion model; ideal observer model; Bayesian modelling; meta-cognition; rational analysis

Introduction

The drift diffusion model (DDM) is a very successful sequential sampling model of the choice process (Ratcliff & McKoon, 2008). Particularly when applied to perceptual decision-making tasks, where the stream of evidence is transparent to the experimenter, this model has shown excellent fits to choice and response time data from a wide variety of experimental paradigms, even generalizing across organisms (Brunton, Botvinick, & Brody, 2013). While it shares several components, including parallel accumulation and race-to-a-threshold with other competing paradigms such as leaky competing accumulation (Usher & McClelland, 2001) and decision field theory (Busemeyer & Townsend, 1993), its stochastic specification of important components of the choice process - rate of accumulation of evidence, response bias, and variability in the evidence accumulation rate - gives it excellent flexibility and interpretability in modelling the summary statistics seen in perceptual decision-making experiments.

While on the one hand, its mathematical construction makes the DDM an excellent *descriptive* model of the choice process, it simultaneously makes it challenging to associate its optimality criterion with the goals and costs faced by real decision-makers. Specifically, the drift diffusion model is known to implement the sequential probability ratio test (SPRT) (Bogacz, Brown, Moehlis, Holmes, & Cohen, 2006), which is statistically optimal in the sense that for any choice of the decision threshold, using the DDM criterion for choice will yield the highest possible accuracy (Wald & Wolfowitz, 1948).

The relationship between the SPRT and DDM imbues it with a normative sense of optimality - observers are being statistically optimal in the SPRT-sense if we show that the DDM model fits their behavior well. But the underlying assumptions of SPRT - perfect evidence integration, approximately linear evidence accumulation rates, race to a fixed evidence threshold - are not good fits for the information and processing limitations that organisms face in real-world decision-making scenarios. In recent years, objections to these premises have been raised on both computational and empirical grounds. Deneve has documented how the conventional drift diffusion paradigm fails to accommodate situations where sensory inputs are unreliable (Deneve, 2012). Thura and colleagues have shown how reaction time distributions in perceptual decision-making tasks may be better described by evidence accumulation terminated by breaching a time-collapsing threshold responsive to an increasing 'urgency' signal than classic accumulation to a fixed threshold (Thura, Beauregard-Racine, Fradet, & Cisek, 2012). Glaze et al have demonstrated how perfect evidence integration - a fundamental assumption of drift diffusion models - is sub-optimal in the face of unsignalled context shifts in the decision-making environment (Glaze, Kable, & Gold, 2015). Thus, using DDM as a normative baseline, research is increasingly focusing on identifying aspects of the environment that constrain real-world decision-making.

Threshold adaptation as metareasoning about effort

We question the premise that decision-makers accumulate evidence up to a fixed threshold. Whereas such proposals have been made previously (Thura et al., 2012), they have focused on incorporating the opportunity cost of continued sampling in the form of a threshold that decreases over time (Drugowitsch, Moreno-Bote, Churchland, Shadlen, & Pouget, 2012). We focus on a different aspect of the threshold determination process - that decision-makers are likely to use information from previous trials to set decision thresholds for upcoming trials. Here again, it is well-documented that patterns of responding can introduce response biases in experiments (Ratcliff & McKoon, 2008). The drift diffusion model allows such biases to be modeled explicitly using changes in diffusion start-point parameters.

We consider a different normative possibility: the threshold parameter used in drift diffusion modeling is a proxy for the amount of effort the observer believes is necessary for adequate performance, and that the observer infers the effort needed for upcoming trials using effort measurements in recent trials. Grounding this hypothesis in a perceptual decision-making task, we model behavior on this task using hierarchical ideal Bayesian observer that performs 2AFC random dot motion (RDM) discrimination, using a meta-

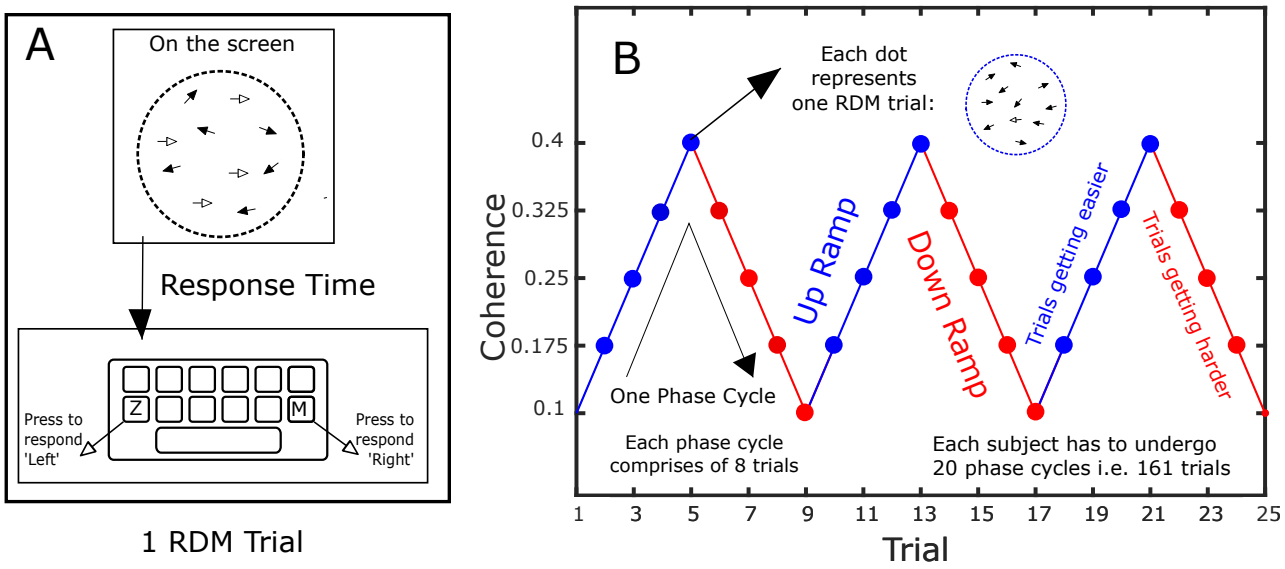


Figure 1: Schematic illustrating the experiment design. (A) On each experiment trial, participants saw a set of dots in Brownian motion, with a horizontal drift added to some fraction of the dots. Participants had to discriminate motion direction using key presses on a computer keyboard, and were incentivized to emphasize accuracy. (B) The sequence of trials each participant saw possessed a higher-order structure, with the difficulty of successive trials increasing and decreasing in a cyclic manner.

cognitive controller to set appropriate values of the decision threshold on each trial. To test this hypothesis, we designed a specific variation of the standard RDM task. In the standard task, trial difficulty is either blocked or randomized across trials. We instead designed a sequence of trial presentation that introduced a predictable trend in the coherence parameter across trials. If people are adaptively tracking the amount of effort they are having to expend on individual trials, we expect such inference to inform their effort allocation on upcoming trials. A hierarchical extension of the drift diffusion model, with a top-down controller setting the evidence threshold adaptively across trials, would potentially fit choice and RT data gathered from such an experiment design better than a simple DDM that assumes a fixed evidence threshold.

Experiment

An RDM Task with higher-order structure

Participants saw a screen with moving dots designed according to the following algorithm. Random motion of the dots was provided by allowing Brownian motion in the vertical direction, i.e. all the dots drifted vertically about their mean position by a distance chosen from a Gaussian distribution. For horizontal movement, we first determined whether a dot was coherent or incoherent as a Bernoulli trial governed by a coherence parameter c. The coherent dots were horizontally moved to a randomly chosen direction (such that each coherent dot was to move in the same direction) by a distance chosen from a Uniform distribution. The incoherent dots were allowed to move horizontally in a randomly chosen direction (such that each incoherent dot had a randomly chosen direction) by a distance chosen from a Uniform distribution.

As in all RDM discrimination experiments, the critical manipulation of task difficulty was governed entirely by the coherence parameter c. We selected a range of values of the coherence parameter by running a calibration pilot with 5 participants, performing 280 trials of the discrimination task under accuracy emphasis. We picked a range of coherence values that permitted 65% accuracy at the low end of the range and 95% accuracy at the high end of the range.

Participants had to indicate the direction of motion of the overall dot pattern on each trial, as illustrated in Figure 1A. They were allowed to take as much time as they wanted to respond to each trial, and as much time as they wanted to rest between the trials. Each correct response fetched points. The scoring system was such that a correct response fetches 10 points; the score of each correct response doubled on responding correctly to three successive trials, and reset to 10 points in case the streak was broken. We encouraged the participants to score well by way of promising a reward to the highest scorer.

The specific higher-order structure introduced in our experiment was a cyclic shift across the 5 specific values of the coherence parameter used in the experiment $\{0.1, 0.175, 0.25, 0.325, 0.4\}$. For example, a participant starting the experiment with a trial with coherence 0.1 would next see a trial with coherence 0.175, then one with 0.25, up to the maximum coherence level of 0.4, beyond which the coherence would begin dropping down to 0.325, then to 0.25 etc. Each such phase cycle from one coherence value through the other 4 and back to the original takes 8 trials, given we use 5 unique coherence values. Participants completed 20 such phase cycles for a total of 161 trials per participant, as illustrated in Figure 1B.

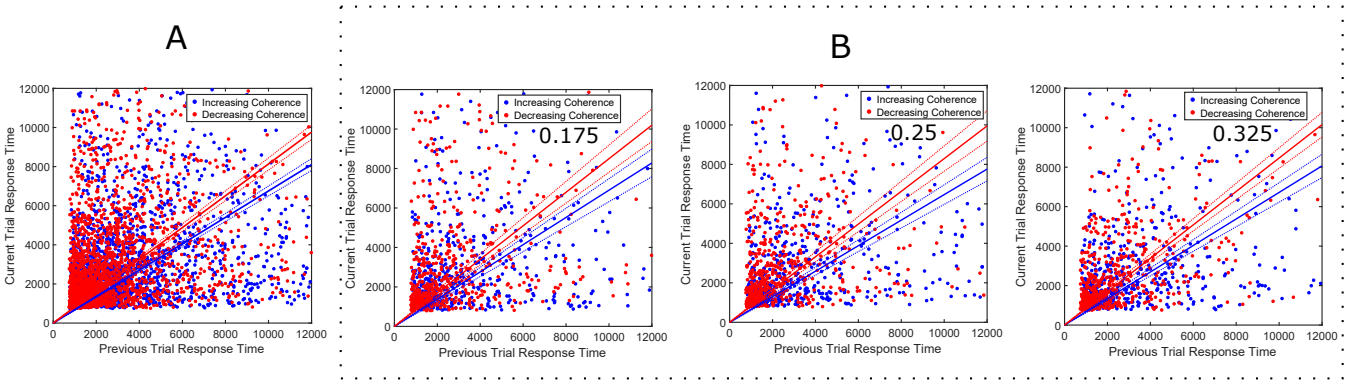


Figure 2: Current trial RT vs Previous trial RT for (A) all coherence levels and (B) individually for each of the three intermediate coherence levels $\{0.175, 0.25, 0.325\}$. Best fit lines along with 95% CI (dotted lines) are shown for both the up ramp and the down ramp. Consistently higher RTs than the preceding trial seen at the same coherence level when the trial difficulty is trending up (i.e. on the down ramp) instead of down suggests adaptive changes in the evidence threshold responsive to past trials.

Task

The task was administered via a web-based interface. Participants indicated responses with keyboard button presses, and were allowed to take as long as they liked before pressing the space bar to begin the next trial.

Since the UI was web-based, there were likely variations in the visual angle across participants. To verify that the visual angles of the display were not beyond our participants' ability to foveate, we calculated the visual display angle for a range of setups based on the following conservative assumptions: (1) the subject used a laptop to access the experiment, (2) the subject was sitting 45-65 cm away from the laptop screen, (3) the laptop used had a screen ranging from 13" - 17" in size, and (4) the laptop used was a model released after 2012. Calculated visual angles for all known standard pixel resolutions for such devices varied from 14 - 28 degrees in the horizontal direction (Mean = 23.4 degrees, SD = 4.4 degrees) , and 8 - 19 degrees in the vertical direction (Mean = 13.9 degrees, SD = 2.7 degrees), mostly within the foveal range of normally sighted observers.

Participants

52 undergraduate and postgraduate students participated in the experiment for course credit (4 female, mean age $20.5 \pm$ 1.57 SD). All participants had normal or normal-corrected vision.

Post-task, we excluded the data for participants who had less than 85% accuracy on the highest coherence trials. For the remaining data (from 34 participants), we excluded the bottom 5% and top 5% response times across the 34 participants for each of the 8 ramp positions to eliminate outliers. We report results both with and without outliers in our analysis below.

Testing the basic prediction

We expected that an observer tracking the cross-trial variations in difficulty would end up tracking the repeated ramp-like movement of the coherence parameter, and take longer on an upcoming trial if the cross-trial coherence was trending downward (i.e. the trials were becoming more difficult) than if it was trending upward (i.e. the trials were becoming easier).

We see that this does happen on average across our $n = 34$ participants (Figure 2A). However, even observers insensitive to cross trial information would be expected to show the same pattern, because upcoming trials aren't just expected to be easier/harder on up/down ramps, they actually are easier/harder too.

Therefore, in Figure 2B we show the current vs previous trial RT stratified by coherence level of the present trial for the three intermediate coherence values in our experiment. By virtue of the up-down phase cycle of our design, for two of the three cases, observers make easy to hard and hard to easy transitions on both up and down ramps. Nonetheless, we still see the same pattern of faster RTs during trials where the coherence had been drifting upward and slower RTs when vice versa which would not happen if observers were insensitive to cross-trial information.

Further, the effect is consistently seen within individuals. In Figure 3 we show the difference between the magnitudes of the slopes for the down ramp and up ramp of the current vs previous trial RT for individual participants. Positive values for all participants confirm that the result holds within individuals in addition to at the cohort level.

These empirical results, while conceptually congruent with our hypothesis, are inadequate to draw strong inferences. We therefore sought to test our metareasoning hypothesis at a trial-by-trial level by developing a generative model of the information accumulation process involved in a 2AFC perceptual learning task that accommodates such adaptive control over the evidence threshold, and comparing its ability to explain our data vis-a-vis a fixed threshold evidence accumulation model.

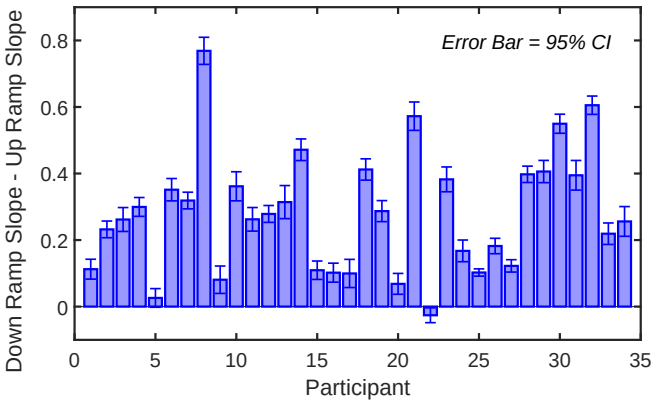


Figure 3: Difference in magnitude of slopes of best fit lines for Up and Down Ramps across all coherence levels for each participant. Consistently higher RTs than the preceding trial when trial difficulty is trending up instead of down suggests adaptive changes in the evidence threshold responsive to past trials for every individual.

A hierarchical drift diffusion model

The DDM, in its standard form is a Wiener diffusion process with drift,

$$dy = vdt + sdW, \tag{1}$$

where y is the diffusion state, v is the drift, s determines the amount of diffusion and dW represents the standard Wiener process. The model accumulates normally distributed pieces of evidence for either alternative until a bound on the cumulative evidence is crossed, and then emits the winning option as the choice.

We developed a hierarchical model of the choice process using a recently proposed Bayesian version of the DDM (Bitzer, Park, Blankenburg, & Kiebel, 2014). This takes the form of a sequential Bayesian update model that maps noisy stimuli observations to latent Gaussian representations

$$x_t \sim N(\mu_i, \sigma^2), \tag{2}$$

constructs a generative model of the likelihood of seeing certain latent feature values for each stimulus alternative,

$$p(x_t|A_i) = N(\hat{\mu}, \hat{\sigma}^2), \tag{3}$$

and updates beliefs about the correctness of a decision alternative given these noisy observations

$$p(A_i|X_{1:t}) = \frac{p(x_t|A_i)p(A_i|X_{1_t-1})}{\sum_{j=1}^{M} p(x_t|A_i)p(A_j|X_{1_t-1})}, \tag{4}$$

where x_t are noisy observations from stimuli belonging to category i, with true prototypical values μ_i and measurement noise σ, estimated prototypical values $\hat{\mu}$ and internal generative variability $\hat{\sigma}$, A_i as possible alternatives and M as the number of considered alternatives. Bitzer et al show that this intuitive ideal observer model is formally identical to the drift diffusion model given certain assumptions about the relationship between parameters of the model.

We augmented this model with a metalearner that estimates the expected sampling effort needed for upcoming trials based on effort allocation on previous trials. We assume a very simple model for this metalearner. It simply updates the effort estimate λ as

$$\lambda_t = \lambda_{t-1} + \gamma\Delta_t, \tag{5}$$

where

$$\Delta_t = z(RT_{t-1}) - z(RT_{t-2}), \tag{6}$$

γ is a free scaling parameter, $z(RT)$ is the normalized z-score of RT at time t with respect to the RT distribution and λ_t serves as the threshold for the DDM for the t^{th} trial. This model is not meant to be comprehensive. We have designed it purely to simulate our expectation of the role of predictable up and down changes in dot motion coherence on observer behavior. We expect that observers will be sensitized to these trends and extrapolate from them to set decision thresholds for upcoming trials. Increasing effort on recent trials should yield a larger threshold for the upcoming trial and vice versa. Normalization is used to induce a natural scale on the size of the change in the threshold; the RT distribution is admittedly non-Gaussian so this assumption could be further refined in future work. Also, to avoid overfitting, we have used the global RT distribution to normalize the RTs, whereas a more realistic model may use sequential summary statistics within participants. Indeed, a filtering-based model might capture the basic intuition of the metalearner more elegantly, but we wished to compare the augmented model with a complicated baseline using only choice and RT data, necessitating parsimony in parameter extension. The version of the meta-learner we have proposed has just one additional free parameter beyond the baseline.

As in (Bitzer et al., 2014), we reduced the set of estimated parameters of the Bayesian model from seven to three by assuming equal amount of drift for both stimuli. In practice we did this by setting $\mu = \hat{\mu} = \pm 1$ for the 2AFC case. Parameter fitting for the 3 parameters to be estimated $\theta = \{\sigma, \hat{\sigma}, t_{nd}\}$ also followed the procedure outlined in (Bitzer et al., 2014). We defined the log likelihood of the data given all parameters as

$$\begin{aligned}\log p(\text{Acc}, \text{RT}|\theta) &= \log p(\text{Acc}|\theta) + \log p(RT|\theta) \\ &\propto -w_{\text{acc}}\ (\text{Acc} - \text{Acc}(\theta))^2 \\ &\quad - \sum_{e=0}^{1}\sum_{i=1}^{7} w_{q_{e,i}}\ (q_{e,i} - q_{e,i}(\theta)^2,\end{aligned}$$

where $q_{e,i}$ is the i^{th} of seven quantiles $(0.02, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9)$ for either correct or error responses as indicated by e.

To evaluate the log likelihood function, again following the procedure in (Bitzer et al., 2014), we simulated our experiment with the Bayesian observer model for different parameter values, averaging the accuracy and RT quantiles obtained

Model	AIC	BIC
Simple DDM	23.2 (0.72)	42.7 (0.72)
Hierarchical DDM	**11.7 (0.47)**	**36.7 (0.47)**

Table 1: Model comparison. Standard deviation across 20 model runs are given in parentheses.

Model	AIC	BIC
Simple DDM	16.5 (2.7)	36.3 (2.7)
Hierarchical DDM	**8.9 (1.5)**	**28.5 (1.5)**

Table 2: Model comparison without outlier removal (i.e. including top 5% and bottom 5% RTs). Standard deviation across 20 model runs are given in parentheses.

across 30 model runs per parameter tuple θ and setting the likelihood weights w to the inverse variance over these repetitions. We scaled each iteration in our simulation to 125 ms. However, we found the MCMC approach advised in (Bitzer et al., 2014) to be too slow (on the order of days) for fitting our hierarchical model that used different threshold values for each trial. Therefore, we used a two-stage grid-search of the parameter space (logarithmic exploration in one stage followed by linear refinement in the second) to optimize the negative log likelihood. In practice, we found that the grid search yields comparable mean parameter values for the baseline DDM model as the MCMC procedure implemented in (Bitzer et al., 2014).

Since we don't use MCMC to obtain a posterior distribution over the parameters, it is useful to average over multiple runs of the likelihood computation at the optimal parameter values to obtain representative likelihood values. After finding optimal parameters via grid search for both models, we calculate model likelihood as the average likelihood obtained from 20 runs of the model for the optimal parameter values. The results from our model comparison using these average likelihood values are presented in Tables 1 and 2. ΔBIC measures difference between baseline DDM and hierarchical DDM BIC. Positive values support the hierarchical DDM, negative values support the baseline model. ΔBIC values with magnitudes smaller than 2 imply insignificant differences between models; values larger than 10 constitute very strong support for a model. Using this measure, the hierarchical DDM is clearly preferable to the simple DDM, across data from all 34 participants (ΔBIC $= 6$ from Table 1 and ΔBIC $= 7.8$ from Table 2).

We additionally ran a block-wise analysis, dividing each participant's trials into 4 sequential blocks of 40 trials and calculating model complexity statistics on the likelihoods emitted by the model for the best fitting parameter values of the overall model ($\sigma = 11, \hat{\sigma} = 8, t_{nd} = 250ms$). We anticipated that any evidence of gradual adoption or relinquishment of threshold metareasoning would show up in the relative model complexity tracked across these four blocks.

As the results in Table 3 show, the hierarchical model is heavily preferred over the simpler model during the first quarter of trials, measured across all participants. For later trials, both models are evenly matched, with the simpler model slightly preferred. This finding is congruent with an account where the participants try to use the higher-order structure across different RDM trials to begin with, but then shift away from it, since it does not offer any material advantage.

Block	1	2	3	4
Δ BIC	12.4	-3.0	-0.07	-0.78

Table 3: Model comparison across four sequential blocks of 40 trials each from all participants. Block 1 contains the first 40 trials from each participant, etc.

Discussion

Researchers have begun to question the drift diffusion model's assumption of a fixed evidence threshold in recent years, basing these arguments on the temporal opportunity costs of continued sampling (Thura et al., 2012). We ask the same question from a different standpoint. We ask whether observers might be sensitive to higher-order statistics in decision-making tasks and adaptively adjust evidence thresholds on upcoming trials to use them efficiently. To see if this can happen, we created a novel variation of the classic random dot motion discrimination task, introducing an up-and-down ramp in difficulty across trials (Gold & Shadlen, 2000). We predicted that observers would be sensitive to this variation. We designed an extension of the drift diffusion model that incorporates metacognitive adaptation of the evidence threshold based on the trend of difficulty of recent trials, and found that it offers a better explanation of participants' behavior in our experiment than a simple drift diffusion model. Our model comparison also suggested initial use and then a shift away from the use of the metareasoning strategy by participants.

Our results support a shift in interpretation of the evidence threshold from its SPRT-driven association with *accuracy*, towards a more general view of it as a metacognitive effort parameter influenced by previous observations. Such a view also makes it easier to generate normative accounts of decisions from memory using DDM-like models, building upon its descriptive success in modeling retrieval success and RT distributions in this domain (Krajbich & Rangel, 2011). Unlike in perceptual decisions where the rate of evidence presentation is fixed, and decisions receive immediate feedback, value-based decisions from memory are made with evidence streams of unknown and unreliable provenance and without feedback. The empirical success of DDM in explaining data from such experiments warrants a broader interpretation of the normative principles of the framework, along the lines proposed in this work.

Previous work has showed that decisions made using either the (log) posterior probability or the log posterior odds

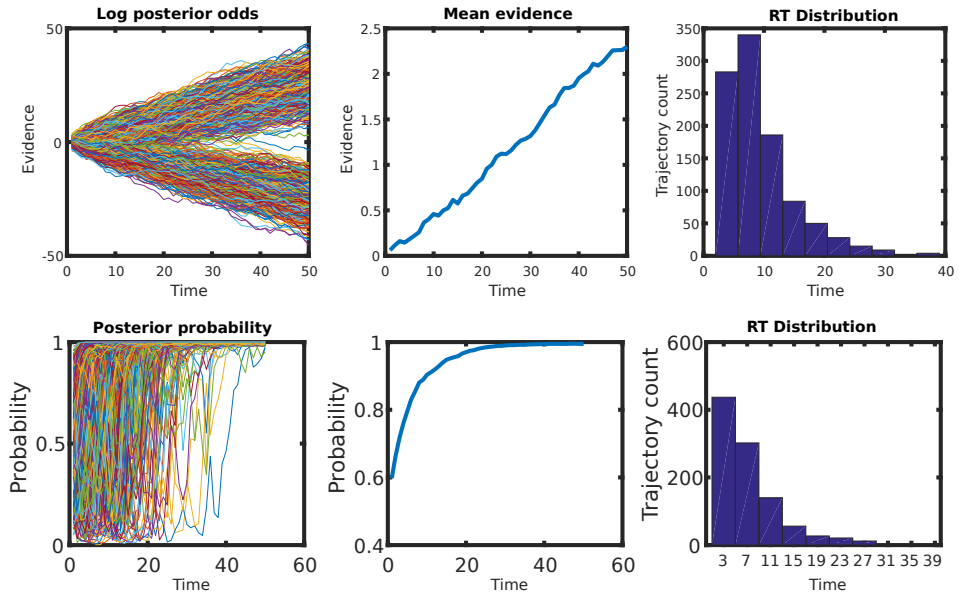


Figure 4: Simulation of the sequential probability ratio test in a 2AFC with noisy Gaussian representations of input stimuli. The log odds ratio rises approximately linearly in the direction of the correct outcome on average (across 1000 simulations) while the posterior probability increases asymptotically towards it (across 1000 simulations). Decision-makers using either decision variable make decisions with very similar accuracy and RT distributions.

yield essentially equivalent observations in terms of accuracy and RT distributions (Bitzer et al., 2014). Conceptually though, the decision-within-a-decision to terminate evidence sampling is vital for decision-makers using the posterior probability, and not the log posterior odds as the decision variable in 2AFC decision-making tasks. Figure 4 demonstrates how the posterior probability, but not the log posterior odds, shows asymptotic sub-linear growth with time on average. A decision-maker using log posterior odds to make decisions can be confident that any threshold he seeks to place on the decision variable will ultimately be breached. A decision-maker setting a bound on the posterior probability of the competing options cannot be so sure. Thus, we suggest the existence of extensive metareasoning about threshold placement indirectly supports the use of the log posterior probabilities instead of log posterior odds (Zhang & Maloney, 2012) as the brain's decision variable in decision-making.

References

Bitzer, S., Park, H., Blankenburg, F., & Kiebel, S. J. (2014). Perceptual decision making: drift-diffusion model is equivalent to a bayesian model. *Frontiers in human neuroscience*, *8*, 102.

Bogacz, R., Brown, E., Moehlis, J., Holmes, P., & Cohen, J. D. (2006). The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological review*, *113*(4), 700.

Brunton, B. W., Botvinick, M. M., & Brody, C. D. (2013). Rats and humans can optimally accumulate evidence for decision-making. *Science*, *340*(6128), 95–98.

Busemeyer, J. R., & Townsend, J. T. (1993). Decision field theory: a dynamic-cognitive approach to decision making in an uncertain environment. *Psychological review*, *100*(3), 432.

Deneve, S. (2012). Making decisions with unknown sensory reliability. *Frontiers in neuroscience*, *6*, 75.

Drugowitsch, J., Moreno-Bote, R., Churchland, A. K., Shadlen, M. N., & Pouget, A. (2012). The cost of accumulating evidence in perceptual decision making. *Journal of Neuroscience*, *32*(11), 3612–3628.

Glaze, C. M., Kable, J. W., & Gold, J. I. (2015). Normative evidence accumulation in unpredictable environments. *Elife*, *4*.

Gold, J. I., & Shadlen, M. N. (2000). Representation of a perceptual decision in developing oculomotor commands. *Nature*, *404*(6776), 390.

Krajbich, I., & Rangel, A. (2011). Multialternative drift-diffusion model predicts the relationship between visual fixations and choice in value-based decisions. *Proceedings of the National Academy of Sciences*, *108*(33), 13852–13857.

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: theory and data for two-choice decision tasks. *Neural computation*, *20*(4), 873–922.

Thura, D., Beauregard-Racine, J., Fradet, C.-W., & Cisek, P. (2012). Decision making by urgency gating: theory and experimental support. *Journal of neurophysiology*, *108*(11), 2912–2930.

Usher, M., & McClelland, J. L. (2001). The time course of perceptual choice: the leaky, competing accumulator model. *Psychological review*, *108*(3), 550.

Wald, A., & Wolfowitz, J. (1948). Optimum character of the sequential probability ratio test. *The Annals of Mathematical Statistics*, 326–339.

Zhang, H., & Maloney, L. T. (2012). Ubiquitous log odds: a common representation of probability and frequency distortion in perception, action, and cognition. *Frontiers in Neuroscience*, *6*, 1.

Mechanisms of Rule Resolution in Premotor Cortex: A Combined TMS/Computational Modeling Study

Patrick J. Rice (pjrice@uw.edu)
Department of Psychology, University of Washington
Campus Box 351525, Seattle, WA 98195 USA

Andrea Stocco (stocco@uw.edu)
Department of Psychology, University of Washington
Campus Box 351525, Seattle, WA 98195 USA

Abstract

In the present study, repetitive transcranial magnetic stimulation was applied over left dorsal premotor cortex (PMd) while participants were performing a novel task paradigm that required on-the-fly planning of responses in accordance with both instructed rules and present stimuli. In conjunction, an ACT-R cognitive model of the task was developed in order to test hypotheses on the potential cognitive functions that may be affected by stimulation of PMd. Two methods of simulating TMS within the ACT-R architecture were tested. In the behavioral experiment, increased response times were observed specifically when TMS was applied while participants were preparing to execute a complex response to an uninstructed stimulus. Model results suggested participant's behavior was due to an effect of TMS on a "re-planning" process, indicating that PMd may be specifically involved in planning of complex motor responses to specific visual stimuli.

Keywords: Rule resolution; TMS; dorsal premotor cortex; ACT-R

Introduction

Rules that guide behavior often consider categories of stimuli and responses; for instance, a particular color may indicate that a specific finger should be used to press a button. Specific identities within these categories are linked through a conditional association to form an exemplar of the rule – following the rule above, a conditional association indicated by the rule may be, "If the light is blue, use your ring finger to respond". This format necessitates representation of rules in an abstract way that allows of consideration of a potentially wide range of stimuli and responses. A growing body of research suggests that abstract rules are represented by frontal cortices. Activation of prefrontal cortex (PFC) has been observed while humans either learn task rules, or execute behavior in accordance with previously-learned rules (Stocco, Lebiere, O'Reilly, & Anderson, 2012).

Despite this, it is obvious that behavior itself is concrete: *specific* effectors are utilized in *specific manners* in response to *specific stimuli*. For effective control, potential/present concrete stimulus identities must be utilized in conjunction with known rules in order to "resolve" a concrete behavioral response. Compared to the representation of abstract rules, relatively little is known regarding this process of rule resolution in the brain. A potential candidate to execute a rule-resolution process is the dorsal premotor cortex (PMd). Anatomically, PMd receives input from both PFC, responsible for abstract rule representation, and posterior parietal cortex, a site of multisensory integration (Tomassini et al., 2007). Electrophysiology conducted in nonhuman primates has revealed that PMd neurons encode movement kinematics (such as direction or amplitude of to-be-performed movements), but that this activity is dependent on the context the movement is performed within (Cisek & Kalaska, 2005). As a result, it has been proposed that PMd may serve to transform simple contextual cues into motor responses (Wise, Boussaoud, Johnson, & Caminiti, 1997).

To investigate whether PMd activity is necessary to resolve behavioral responses in accordance with known rules and stimuli, we constructed a novel task based on the Rapid Instructed Task Learning (RITL) paradigm, which allows for the examination of encoding, planning, and execution of rules (Cole, Laurent, & Stocco, 2013). On each trial, the participants were first required to encode a trial-specific conditional association, and then resolve and execute a response after a stimulus was given. Rules that generated conditional associations were categorized as either "Concrete", in which a specific stimulus identity (e.g., a number being "even") was associated with a specific motor effector (e.g., index finger), or "Symbolic", in which a specific stimulus identity was associated with a placeholder symbol that, upon stimulus presentation, allowed resolution of a secondary association that indicated a specific response. Stimuli on a subset of trials were designed to violate the instructed conditional association, requiring participants to replan the correct response.

While performing this task, high-frequency repetitive transcranial magnetic stimulation (rTMS) was applied over PMd during one of two time points during a given trial: either while presenting the trial-specific conditional association (termed "early"), or while presenting the stimulus to respond to ("late"). If PMd is involved in the encoding of the rule, rTMS should significantly affect response times during the "early" encoding phase. Alternatively, if PMd underpins rule resolution, "late" rTMS should leading to increased response times during the execution phase.

Interpretation of the effects of TMS on behavior requires a careful consideration of the possible processes that might be disrupted by the TMS pulse train. A high-frequency pulse has limited temporal duration (in our experiment, 0.5 seconds) and its effects crucially depend on the timing of the underlying process. For example, a delay in the response to

a stimulus could be interpreted as either resulting from an inhibitory effect on motor execution or from an inhibitory effect on the retrieval of a stimulus-response mapping *before* the execution. Without a reasonable process model, it is impossible to fully examine the space of possible explanations. For this reason, the effects of rTMS on behavior were also compared to the predictions of a computational model developed within the ACT-R cognitive architecture (Anderson, 2007)[1]. ACT-R is the most successful and popular cognitive architecture, and allows for the testing of specific hypotheses of cognition in a controlled and parameterized manner, as the architecture's constituent components are linked to known brain regions and functions. ACT-R models are a combination of declarative, factual knowledge and procedural knowledge. Declarative knowledge is stored in the form of structured records ("chunks"), while procedural knowledge is represented as IF-THEN rules (production rules or "productions"). Behavior unfolds as one production at the time is selected for execution, and it retrieves, moves, or modifies chunks as part of its actions. Chunks and productions interact with each other through a series of "buffers", limited-capacity temporary stores that hold a single chunk for extended periods of time, making it accessible to productions.

We reasoned that, between the concrete and symbolic rule conditions, identical buffers and production rules should be utilized to produce behavior under the task. In general, the model utilizes the presentation of the conditional association to "plan" a response to the specified stimulus. When presented with a stimulus, if it matches that which was planned for, the response is immediately executed. However, if there is a violation of the expected stimulus, a "re-planning" occurs in which a new plan considering the evident stimulus is prepared and subsequently executed. The only difference between how the model considers "concrete" compared to "symbolic" conditions is the information content of the chunk produced by "planning". We hypothesized that the "re-planning" in response to a violation of expected stimulus identity is what may be affected by rTMS in the behavioral experiment. To our knowledge, the effects of TMS on brain and behavior have never been simulated in ACT-R, and ACT-R has never been used to clarify the possible interpretations of TMS effects.

Methods

Participants

Twelve right-handed participants (8 females, mean age = 24.7 $\pm$ 3.3) with no history of neurological disorder, head injury, or any other contraindications to rTMS were were recruited to participate in the study. Recruitment was restricted to individuals who had previously participated in neuroimaging experiments at the University of Washington, and for whom structural and functional imaging data already existed. All participants received monetary compensation proportional to the total amount of time devoted to the study.

Task Paradigm

We constructed a RITL-based paradigm focused on conditional motor behaviors. The progression of the task is depicted in Figure 1. Participants were instructed to determine the parity of a numeric stimulus (restricted to the single-digit numbers 2 – 9; presented in the center of the screen during stimulus presentation) and respond on the basis of a trial-specific rule. Responses occurred by pressing the "left" or "right" arrow keys on a standard keyboard with the participant's right-hand index and middle fingers, respectively.

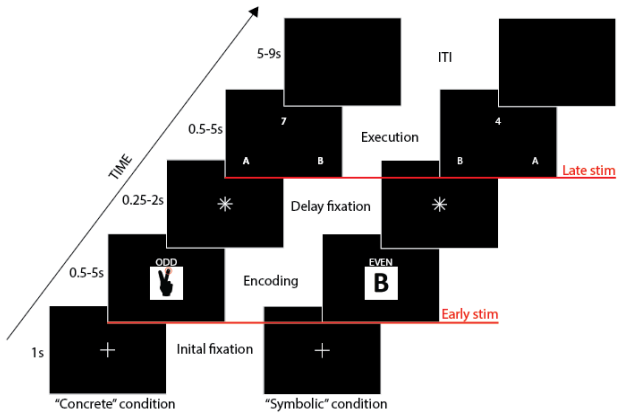


Figure 1: Time course of concrete (left) and symbolic (right) trials. Red lines represent the moments at which rTMS was applied.

Participants were presented with two types of rules: "Concrete" rules, which indicated the association of a specific effector (either index or middle fingers of the right hand) to a specific parity ("EVEN" or "ODD"); and "Symbolic" rules, which indicated the association of a specific letter on the screen ("A" or "B") to a specific parity. Specific effectors in the "Concrete" condition were indicated during rule presentation by a stylized hand (black on white background), with the rule-specific effector denoted by a red circle around the tip of the finger. Specific effectors in the "Symbolic" condition were indicated during stimulus presentation by the placement of the letters "A" and "B", which were randomly assigned to the bottom left and right corners of the screen on a trial-by-trial basis. To make the two conditions visually comparable, these letters appeared during the stimulus presentation phase of both "concrete" and "symbolic" trials, although they only carried meaning in the "symbolic" condition. Participants were informed that the bottom left corner corresponded to the "left" arrow key, while the bottom right corner corresponded to the "right" arrow key. Due to this manipulation, during a "symbolic" trial participants could not prepare a specific motor effector until stimulus presentation.

[1]The model code, together with the experimental data, can be found at our laboratory's GitHub account: http://github.com/UWCCDL/DRI/

Crucially, instructions specified only half of a trial's rule. That is, participants may be given the instructions "*EVEN*:Index" but then asked to respond to a stimulus (e.g., "7") that is *odd*. Under these circumstances, participants have to mentally re-plan a new stimulus-response configuration. This created a third experimental factor in which the trial is either "instructed" (e.g., the instructions mention an *even* number, and the stimulus is *even*) or "inferred" (i.e., the instructions mention an *even* number, but the stimulus is *odd*). Inferred trials are of particular interest because the re-processing of the rule likely occurs within the window of delivery of a TMS pulse train during "late" stimulation.

The task consisted of 4 blocks of 60 trials each, with a 5-minute break enforced between each block. At the beginning of each trial, a central fixation cross was illuminated for 1s. Once the fixation was extinguished, the rule informing the subject of the individual trial's valid associations was displayed for a maximum of 5s. Presentation of rule condition was randomized across trials within a block. Subjects acknowledged understanding of the presented rule by pressing the spacebar with their *left* hand, so as to not to interfere with the activity of the *right* hand used for the response mappings. After subjects acknowledged the rule, a variable (0.25 – 2 seconds) delay occurred while a fixation "asterisk" was displayed (Figure 1). Once this delay had passed, the stimulus was displayed and subjects were given a 5 second window to respond by pressing the left or right arrow key with the index or middle finger of their right hand. Upon response (or after 5 seconds had passed), a variable inter-trial interval (5 – 9 seconds) was enforced while a blank screen was displayed.

Event-related, high-frequency rTMS was delivered across two sites (left PMd, experimental; Vertex, control) in alternating blocks, with the order counterbalanced across subjects. Commonly used as a control condition in TMS studies, Vertex stimulation provides the same scalp sensation and audible click as stimulation of the targeted region, but does not evoke a functionally significant neural response (Jung, Bungert, Bowtell, & Jackson, 2016). As noted above, there were two possible time points of stimulation during a trial, either upon presentation of the rule instructions ("early" stimulation), or upon presentation of the stimulus ("late" stimulation; Figure 1). In one third of trials in a given block, no stimulation was delivered; in all other trials, only one stimulation train occurred (either "early" or "late"). Trials were pseudo-randomized so that consecutive series of three trials each contained one instance of early stimulation, one instance of late stimulation, and one instance of no stimulation. Due to the possibility of rTMS delivery either inducing an unwanted motor response or suppressing a genuine response, subjects were locked out of responding to either screen for the first 0.5 seconds of presentation, and made aware of this fact. In agreement with the rTMS safety guidelines (Rossi, Hallett, Rossini, & Pascual-Leone, 2009), instances of stimulation did not occur more than once every 8 seconds.

Transcranial Magnetic Stimulation

Parameters High-frequency, event-related rTMS was applied with a 70–mm figure–of–eight coil (Double 70–mm Alpha coil, Magstim, UK) connected to a biphasic transcranial magnetic stimulator (Super Rapid^2, Magstim, UK). Each stimulations consisted of a five-pulse train at 10 Hz and 110% of the individual's Resting Motor Threshold (RMT). The coil was placed over the stimulation sites tangential to the skull, with the handle pointed at 45° to the sagittal plane (PMd) or parallel to the midline (Vertex).

Assessment of Resting Motor Threshold Each participant's RMT was estimated using motor evoked potentials (MEPs) elicited in response to stimulation of the primary motor cortex. MEPs were estimated from electromyography surface electrodes on the right first and third dorsal interosseous (FDI) muscles in a belly-tendon montage. Valid MEPs ($>$ 50 mV peak-to-peak amplitude) were used to calibrate the TMS intensity using a sequential estimation algorithm (PEST: Taylor and Creelman, 1967).

Target Localization for TMS The participants' existing structural and functional MRI data were used to achieve submillimeter precision in the targeting of stimulation sites. The position of the TMS coil was controlled through a frameless stereotaxic system, which tracked the location of the participant's heads relative to the coil with an infrared tracker camera and co-registered these locations with the individual participants structural and functional images. The location of PMd stimulation was determined by targeting the most significant voxels of a cluster identified while the subject was making a motor response with their fingers, ensuring that the location was in good agreement with published anatomical landmarks. The Vertex was located over the sagittal midline, at the level of the postcentral gyri.

Computational Model

The ACT-R model consists of 18 production rules and 30 chunks. The model's strategy is illustrated in the flowchart of Figure 2, and follows common assumptions. In broad strokes, during each trials the model proceeds through five consecutive stages: (1) Instruction encoding, (2) Task plan preparation, (3) Parity judgment, (4) Task re-planning (if necessary), and eventually (5) Motor response.

The task instructions are first encoded by storing the parity ("EVEN" or "ODD") and the action (finger: "Index" or "Middle"; or letter: "A" or "B") into a working memory buffer. Following this shallow encoding, the model uses the information to prepare a mental plan of the action to perform (a finger movement for concrete rules, or a visual search followed by a finger movement for symbolic rules) if the parity of the stimulus matches that which was instructed. The mental plan contains a detailed chunk specification of the stimulus quality to verify (i.e., the parity of a number) and the action to perform (the motor action or the visuo-motor procedure), and is thus specific to the type of rule (as underscored by the

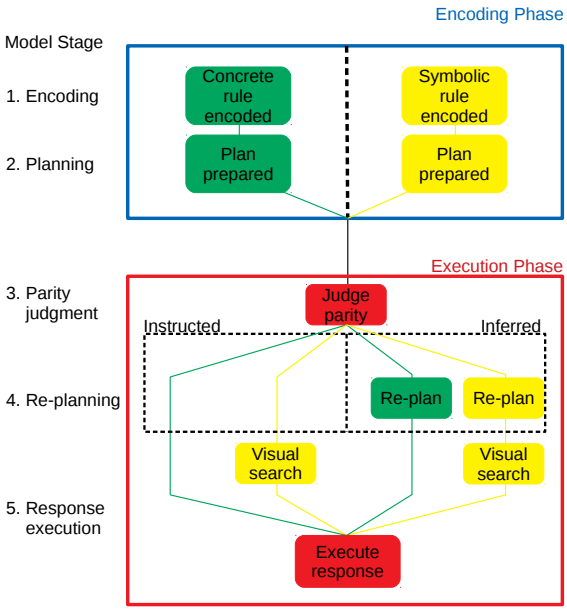


Figure 2: Model's processing strategy and stages

yellow and green boxes in Figure 2). There is ample evidence in the RITL literature for the existence of such intermediate, deeper representations of the task (Cole et al., 2013; Stocco et al., 2012). After planning is completed, the model uses the left hand to proceed to the execution phase.

During the *execution* phase, the model first retrieves the parity associated with the stimulus (i.e., the declarative fact that "8 is EVEN"). If the parity of the stimulus matches the parity considered by the previously prepared plan, the model proceeds with the response phase. If the parity of the stimulus does *not* match the condition for the intended action, as in the case of "Inferred" trials, the model discards and re-prepares the mental plan on the basis of the apparent parity of the stimulus. This re-planning is again specific to the type of rule (green and yellow "Re-plan" boxes in Figure 2). After re-planning, the model proceeds to the response phase, in which the model simply follows the most recently prepared plan and executes the visuo-motor commands as specified.

Modeling the Effects of TMS There are no established guidelines nor precedents on how to model the effects of high-frequency rTMS in ACT-R. For this reason, two complementary methods in which TMS stimulation could be modeled within the ACT-R architecture were considered:

- **Method 1**: TMS affects the operation of individual ACT-R *buffers*. This is consistent with the established ACT-R literature, in which buffers are associated to different cognitive modules and correspond to distinct cortical regions specialized for different functions (Anderson, Fincham, Qin, & Stocco, 2008).
- **Method 2**: TMS affects the execution of individual ACT-R *productions*. Although the canonical interpretation is that ACT-R productions are related to the basal ganglia (Anderson et al., 2008), it has been argued in the past that productions can alternatively be interpreted as cortico-cortical connections (Stocco, Lebiere, & Anderson, 2010). Thus, since productions combine knowledge from different sources, they might be ideally suited to represent the function of cortical associative areas, such as PMd.

In both cases, the effects of TMS were simulated by forcing a delay into the operation time of the corresponding target structure (buffer or production), with the delay extending until the end of the simulated rTMS pulse train. For Method 1, this was achieved by forcing a buffer's *status* to "failure" throughout the duration of the TMS train. For Method 2, this was achieved by modifying a production's *action time* parameter at the moment of selection.

Experimental Results

Participants were highly accurate throughout the task (M = 95%), with no significant differences in accuracy due to either experimental manipulation or TMS. For this reason, the analyses reported here will focus on the response times.

In both the encoding and execution phases, there was no difference in response times between trials belonging to either of the two control conditions (Vertex stimulation and no stimulation trials). Thus, the No-stimulation trials were excluded from the remaining analyses, so that comparisons were always made between conditions in which the participants received stimulation (on Vertex and PMd sites, respectively) .

The main behavioral results of our experiment are summarized in Figure 3. In the encoding phase, a two-way ANOVA considering the effect of site of stimulation and type of rule revealed no significant main effects or interactions of these conditions on the encoding response time. However, in the execution phase, a four-way ANOVA examining the effect of site of stimulation, timing of stimulation, type of rule, and instruction/inference of rule on response times revealed a main effect of rule ($F(1,7) = 238.3$, $p < 0.0001$, $\eta^2 = 0.12$) alongside a main effect of instruction/inference ($F(1,7) = 31.44$, $p = 0.0008$, $\eta^2 = 0.061$). A significant two–way interaction between site of stimulation and type of rule was observed ($F(1,7) = 16.7$, $p = 0.0047$, $\eta^2 = 0.006$), while a significant three–way interaction between site of stimulation, timing of stimulation, and instruction/inference ($F(1,7) = 13.3$, $p = 0.0082$, $\eta^2 = 0.006$) was also present. Subsequent two-way ANOVAs considering site of stimulation and type of rule, performed within the subconditions defined by the conjunction of timing of stimulation and instruction/inference, demonstrated that the Inferred-Late stimulation subcondition was the only subcondition in which the two–way interaction between site of stimulation and type of rule occurred ($F(1,7) = 10.11$, $p = 0.015$, $\eta^2 = 0.037$). These effects were qualitatively similar across the eight participants considered by the analysis. Within the Inferred-Late stimulation subcondition, paired t-tests revealed there to be no difference in mean re-

sponse times between PMd and Vertex stimulation on "Concrete" trials (paired $t(7) = 0.119$, $p = 0.909$, Cohen's $d = 0.08$), but a significant difference in mean response times between PMd and Vertex stimulation on "Symbolic" trials was observed (paired $t(7) = 3.21$, $p = 0.015$, Cohen's $d = 2.27$).

In summary, rTMS on PMd yielded a very specific effect, significantly delaying response only in the execution of symbolic rules, and only when the rules need to be re-processed ("Inferred" trials: red column in Figure 3C). These results strongly suggest an involvement of PMd during the creation of mental plans for "Symbolic" rules only. This interpretation was further examined through the ACT-R model.

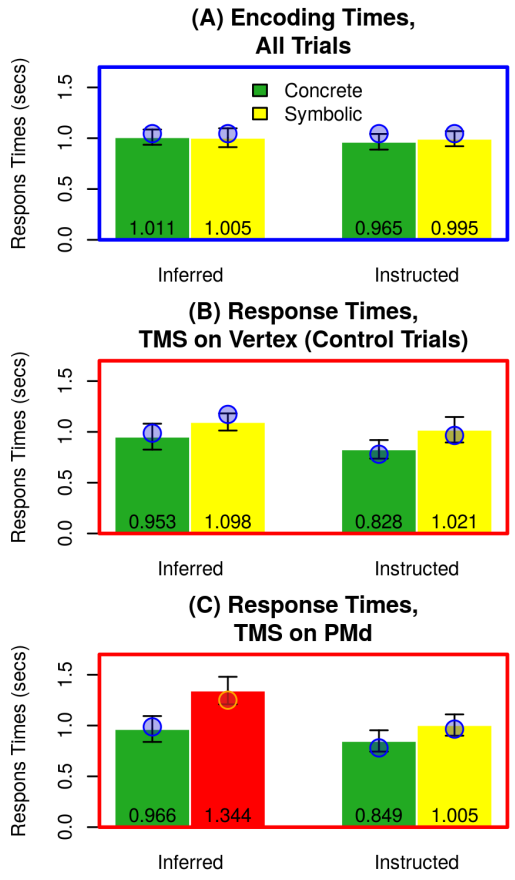


Figure 3: Experimental results and model simulations. Columns represent mean values +/- SEM. Green denotes concrete trials; yellow denotes symbolic trials; red denotes the effects of rTMS. Circles represent model predictions.

Model Results

Before examining the effects of TMS, the model was fit to behavioral data from the control conditions. The process of fitting the model required a minimal amount of tuning, and concentrated on the three parameters: (1) the imaginal latency L, which determines the time to create or update the mental plan; (2) The base-level activation B of the parity facts that are retrieved; (3) and the motor preparation time M, which determines the time to prepare a motor response from instructions. The parameter L can be estimated from the experimental data, as the only difference between the execution of "Inferred" and "Instructed" rules is the time necessary to update the mental plan. In our experimental data, this difference was 133 ms. In the model, this difference corresponds to the time to execute a production that updates the imaginal buffer, and is the sum of the production's action time (50 ms) and L, which can then be estimated as 133 - 50 = 83 ms. The values of $B = 1.5$ and $M = 150$ ms were estimated through a grid-search procedure to minimize the discrepancy between the model's predicted mean reaction times and the mean reaction times of the experimental data.

When fit to the empirical behavioral data, the model provides predictions of participant performance without stimulation. The model's performance is represented by the blue dots in Figure 3A–C. Without simulated stimulation, the model reproduces three key patterns in participant behavior. The first is that both type of rules take the same time to encode (Figure 3A: $RMSE$ = 64ms, $\chi(3) = 1.43$, $p > 0.69$). The second is an increased response time for "Symbolic" trials relative to "Concrete" trials (compare green to yellow bars in Figure 3B). The third is an increased response time for Inferred trials relative to Instructed trials (compare yellow or green bars in Figure 3B: $RMSE$ = 55ms, $\chi(3) = 3.27$, $p > 0.35$). This is a good indication that the model implementation is representative of the cognitive dynamics required by the task.

After parameter fitting, the two methods of TMS simulation were applied exhaustively to all buffers (Method 1) and productions (Method 2) of the model. A parameter space partitioning approach was applied to the simulations to identify the conditions in which the model produced the qualitative pattern of Figure 3C, that is, a significant effect of TMS only for the Inferred Symbolic trials. Interfering with the operation of a given buffer (i.e., Method 1 of TMS simulation) could never reproduce the results observed in the behavioral experiment, as the simulated interference always caused increased response times across multiple subconditions (dependent on the exact buffer that is manipulated). In comparison, Method 2 was capable of replicating the behavioral result in one specific case–when the production responsible for re-planning a symbolic trial was delayed (i.e., the production corresponding to the yellow "re-plan" box in Figure 2). In this condition, the model displayed increased response times specifically for the Symbolic-Inferred-Late subcondition (the red bar in Figure 3C: $RMSE$ = 62ms, $\chi(3) = 1.64$, $p > 0.65$). Disruption of any other production could not reproduce this effect, providing greater support for our conclusion that the behavioral result is explained by assuming a role of PMd in the preparation of complex responses that require visuomotor coordination.

Discussion

Application of rTMS during a RITL paradigm revealed that PMd was specifically involved when participants had to re-

solve an inferred conditional rule that was abstracted away from concrete response identities, as demonstrated by increased response times in the Symbolic-Inferred-Late stimulation subcondition. In comparison, when the conditional association had been instructed, or when it considered concrete response identities, rTMS over left PMd had no significant effect relative to controls. To investigate the potential cognitive operations that were affected by rTMS (and therefore potential cognitive operations performed by PMd), we developed an ACT-R cognitive model of the task. The model performs the task by creating a "response plan" on the basis of the trial-specific instructions, thus preparing to respond when a valid stimulus appears. If there is a mismatch between the expected and actual stimulus, a "re-planning" occurs in which the actual stimulus parity is used to create a new response plan to guide response execution. The effect of TMS was simulated through both a buffer-specific method and a production-specific method.

The model reproduced the two major patterns of participant behavior under this task: an increase in response time for "Symbolic" relative to "Concrete" trials, and an increase in response time for "Inferred" trials relative to "Instructed" trials. When simulated TMS was applied to the model's buffers (Method 1), the model could not reproduce the main behavioral result (i.e., a specific effect of TMS within the Inferred-Symbolic-Late stimulation subcondition). Method 2 of TMS simulation was capable of replicating the behavioral results, specifically when the production dedicated to re-planning of "Symbolic" response plans was targeted. This finding, together with the behavioral data, provides strong evidence of an involvement of PMd in the planning of complex responses to specific stimuli. From a computational point of view, the model suggests that the function of brain regions not traditionally associated with ACT-R buffers (such as PMd) might be interpreted in terms of specific *productions*, whose operations represent cortical associative areas.

The use of a plausible computational model provides a new way to predict the effects of stimulation. Examination of the model's structure allows for *a priori* predictions of participant behavior in response to varied regimes of TMS application. For example, since the model encodes the task instructions *before* creating a response plan, "early" stimulation (i.e., time-locked to presentation of the encoding phase) of the productions responsible for the initial "planning" does not change the model's behavior. If, however, stimulation were to instead occur later in the encoding phase, the model predicts that response planning would be disrupted and encoding times during Symbolic-Instructed trials would be increased (opposite to what was observed in the present experiment). Another prediction is that a different cortical region, distinct from PMd, is responsible for planning and re-planning "Concrete" responses (green boxes in Figure 2). A likely candidate for this cortical region is the ventral premotor cortex (PMv), a brain area (shown to be anatomically distinct from PMd) which demonstrates functional selectivity for concrete motor actions (Rizzolatti, Fogassi, & Gallese, 2002). Taken together, these results suggests that computational models and TMS are complementary tools for cognitive neuroscience research.

References

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

Anderson, J. R., Fincham, J. M., Qin, Y., & Stocco, A. (2008). A central circuit of the mind. *Trends in Cognitive Sciences*, *12*(4), 136–143.

Cisek, P., & Kalaska, J. F. (2005). Neural correlates of reaching decisions in dorsal premotor cortex: specification of multiple direction choices and final selection of action. *Neuron*, *45*(5), 801–814.

Cole, M. W., Laurent, P., & Stocco, A. (2013). Rapid instructed task learning: A new window into the human brains unique capacity for flexible cognitive control. *Cognitive, Affective, & Behavioral Neuroscience*, *13*(1), 1–22.

Jung, J., Bungert, A., Bowtell, R., & Jackson, S. R. (2016). Vertex stimulation as a control site for transcranial magnetic stimulation: A concurrent TMS/fMRI study. *Brain Stimulation*, *9*(1), 58–64.

Rizzolatti, G., Fogassi, L., & Gallese, V. (2002). Motor and cognitive functions of the ventral premotor cortex. *Current Opinion in Neurobiology*, *12*(2), 149-154.

Rossi, S., Hallett, M., Rossini, P. M., & Pascual-Leone, A. (2009). Safety, ethical considerations, and application guidelines for the use of transcranial magnetic stimulation in clinical practice and research. *Clinical neurophysiology*, *120*(12), 2008–2039.

Stocco, A., Lebiere, C., & Anderson, J. R. (2010). Conditional routing of information to the cortex: A model of the basal ganglias role in cognitive coordination. *Psychological Review*, *117*(2), 541.

Stocco, A., Lebiere, C., O'Reilly, R. C., & Anderson, J. R. (2012). Distinct contributions of the caudate nucleus, rostral prefrontal cortex, and parietal cortex to the execution of instructed tasks. *Cognitive, affective, & behavioral neuroscience*, *12*(4), 611–628.

Taylor, M., & Creelman, C. D. (1967). Pest: Efficient estimates on probability functions. *The Journal of the Acoustical Society of America*, *41*(4A), 782–787.

Tomassini, V., Jbabdi, S., Klein, J., Behrens, T., Pozzilli, C., Matthews, P., ... Johansen-Berg, H. (2007). Diffusion-weighted imaging tractography-based parcellation of the human lateral premotor cortex identifies dorsal and ventral subregions with anatomical and functional specializations. *Journal of Neuroscience*, *27*(38), 10259–10269.

Wise, S. P., Boussaoud, D., Johnson, P. B., & Caminiti, R. (1997). Premotor and parietal cortex: corticocortical connectivity and combinatorial computations. *Annual Review of Neuroscience*, *20*, 25–42.

Implications of Guessing Types in Multinomial Processing Tree Models: Conditional Reasoning as an Example

Nicolas Riesterer (riestern@cs.uni-freiburg.de)
Cognitive Computation Lab, Georges-Koehler-Allee 79
79110 Freiburg, Germany

Marco Ragni (ragni@cs.uni-freiburg.de)
Cognitive Computation Lab, Georges-Koehler-Allee 79
79110 Freiburg, Germany

Abstract

Human responses in reasoning are sometimes based on guessing which is a cognitive process usually accounted for by adequate cognitive models. In the literature, different types of guessing have been reported but investigations of their impact on the overall model performance are rare.

This article focuses on three theories of conditional reasoning implemented as Multinomial Processing Trees (Oberauer, 2006). We analyze the impact of the different guessing types on the raw goodness of fit, on information criteria commonly found in the literature (AIC, BIC, FIA), discuss the partial influences of reasoning, guessing, and additional heuristic components, as well as assess the impact of guessing on parameter estimates. The results indicate that using different types of guessing can have a reliable impact on the model's performance and implications about the assumed cognitive processes.

Keywords: Cognitive Modeling; Conditional Reasoning; Multinomial Processing Trees; Guessing

Introduction

Computational modeling has recently gained popularity in a wide variety of research domains for its capability to evaluate and compare competing theoretical accounts in a well-defined manner. In cognitive science and psychology, theories are now routinely formalized as computational models such as *Multinomial Processing Trees* (MPTs; Riefer & Batchelder, 1988; Purdy & Batchelder, 2009). These models are of particular interest due to their inherent capability of modeling latent processes and have been used in a multitude of research domains such as memory storage and retrieval, perception, reasoning, or psychometrics (e.g., Batchelder & Riefer, 1999; Erdfelder et al., 2009). Even in the case of underspecified or verbally defined theories, they allow for a profound assessment of underlying assumptions.

Computational modeling is comprised of three phases: *model implementation*, *model evaluation*, and *model comparison*. While methods for model evaluation and comparison are standardized and available in various toolboxes (e.g., MPTinR; Singmann & Kellen, 2013), the development of models still offers considerable freedom to the modeler. In principle, there are (at least) two computational modeling approaches: First, cognitive computational models are developed to formalize (verbally specified) cognitive theories to evaluate their explanatory power (e.g., Oberauer, 2006). Second, the connections between identified cognitive processes and experimental variables can be systematically manipulated to develop a cognitively grounded theory (e.g., Klauer, Stahl, & Erdfelder, 2007).

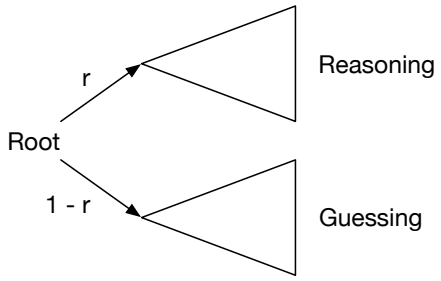


Figure 1: General structure of MPT models augmented with a guessing subtree. A parameter *r* represents the probability of entering the reasoning submodel, i.e., the process motivated by a cognitive theory.

Different goals have different requirements with respect to the modeling approaches being applied. Theory formalization requires the model to be as close to the original theoretical specification as possible. This largely prevents the modeler from incorporating additional assumptions targeted towards optimizing the performance of the resulting model. On the other hand, in general cognitive modeling, the modeler might decide on modifying the original theory in order to account for missing outcomes, or to make the resulting model comparable to a selection of alternative models with incompatible high-level process assumptions.

A model augmentation commonly found in the literature is a process usually coined "guessing". While technically representing a response generation process differing from regular inference, guessing is often incorporated into models as an alternative strategy to account for missing response categories in order to enable evaluation on general datasets or comparison with other candidate models (cf. Figure 1). Guessing can be based on different methods such as *uniform* approaches assigning equal probability to all outcomes, *bias* guessing assuming a bias for general acceptance of conclusions, or *independence* guessing specifying independent bias probabilities for individual inferences. Even though including a guessing component is a common asset in cognitive modeling, discussions of its influence and potential for negatively influenc-

ing obtained results are rarely found. Furthermore, previous research showed that sometimes guessing alone can achieve better performance than when paired with a reasoning part (Ragni & Tse, 2017). While this previous study evaluated the contribution of each inference pattern for conditional reasoning (with a bias guessing approach), the overall impact of different types of guessing has not been systematically analyzed yet.

Our goal is to add to the comprehensible and in-depth analysis conducted by Oberauer (2006). We extend it by analyzing the impact of different types of guessing on raw model performance with respect to goodness of fit, and a set of established information criteria (AIC, BIC, FIA), assessing the impact of guessing and a heuristically driven system, as well as evaluating the influence of guessing on parameter estimates. While we use the term *guessing* consistent with the literature in referring to the additional subtree, this does not imply that it can only reflect guessing processes and not alternative strategies such as heuristics. Our analysis compares the effects of three types of guessing: random guessing, a heuristically guided process and a theoretically motivated approach.

The article is structured as follows: In the next section we present our demonstrative domain of conditional reasoning and introduce three theories implemented as MPTs by (Oberauer, 2006). In Section 3, this set of models is augmented with different types of guessing and fit to data in order to analyze the impact of guessing on model performance and individual parameter estimates. A discussion about the impact of guessing types concludes the article.

State of the Art

In this section we briefly introduce conditional reasoning as our demonstrative domain and sketch the set of cognitive theories we base our analysis upon (Oberauer, 2006).

A conditional such as *if A then C* consists of an antecedent (here abbreviated by *A*) and a consequent (here abbreviated by *C*). If additional knowledge is given, such as *A*, then the consequent *C* can be inferred from the conditional (*Modus Ponens*, MP), for other additional information, such as not-*C*, the inference mechanism *Modus Tollens* (MT) can be applied to infer not-*A*. Both mechanisms are correct with respect to propositional logic. If instead the information *C* is given, the *Affirmation of Consequent* (AC) is sometimes applied by reasoners to infer *A*. If not-*A* is given the *Denial of Antecedent* (DA) can be applied to infer not-*C*. The last two mechanisms only make sense if the conditional is interpreted as a biconditional.

Human reasoners do deviate from the classical logical inferences (e.g., Klauer et al., 2007) and so cognitive theories have been developed that can better capture the human inference process than purely logical systems. Among them are the *Mental Model Theory* (MMT; Johnson-Laird, 1990; Johnson-Laird & Byrne, 2002) that assumes that human reasoners do reason with respect to a mental model of a conditional. A mental model is an interpretation of the given conditional with the initial mental model assuming that the antecedent and consequent are both true at the same time. While performing the reasoning task, other interpretations are possible and can be derived in a flesh-out process.

For his investigation of conditional reasoning, Oberauer (2006) implemented a set of models as *Multinomial Processing Trees* (MPTs; Riefer & Batchelder, 1988; Purdy & Batchelder, 2009). MPTs are a family of probabilistic models which can be used to model categorical data. Their core principle is based on the multinomial distribution and the assumption that observations are the product of a series of latent processes. By defining a tree structure on the set of assumed latent cognitive processes, MPTs can be used to test hypotheses related to human cognition. Model comparison is usually performed on the basis of so-called *Information Criteria* weighing the complexity of a model against the achieved fit to experimental data. The most prominent information criteria are *Akaike's Information Criterion* (AIC; Akaike, 1974), the *Bayesian Information Criterion* (BIC; Schwarz, 1978), and the Fisher Information Approximation (FIA; Rissanen, 1996).

Oberauer (2006) developed two models following the mental model theory as MPTs: one with directionality (i.e., with an additional assumption that mental models are processed from the antecedent to the consequent) and one without this additional assumption. These models define predictions for individual patterns on the basis of processes underlying the MMT account of reasoning, i.e., he presents MPTs that can predict an individual reasoners' choice of inference mechanisms from MP, AC, DA, MT.

Apart from MMT, Oberauer (2006) presents three further implementations of cognitive theories in form of MPTs: two variations of the *Suppositional Theory* (Evans & Over, 2004) which assumes that there are two reasoning processes, a fast and heuristic and a slower analytic one, as well as a *Dual-Process* (DP) approach (Verschueren, Schaeken, & d'Ydewalle, 2005) which combines inference on the basis of heuristic probability estimation with MMT.

Analysis of Guessing Types

Method

Guessing is understood as a way to produce answers "when reasoners are uncertain about the appropriate response but have to make a response nevertheless" (Klauer et al., 2007). In this sense, guessing represents an alternative strategy to reasoning for producing responses. It serves the purpose of explicitly representing human uncertainty as well as a general means of producing answers not accounted for by the theoretical account for reasoning.

Technically, guessing is a distribution over all possible responses. Depending on its degrees of freedom, it can represent true random guessing, or alternative strategies which may include biases or other strategies of varying complexity. In the following we investigate the effects of guessing based

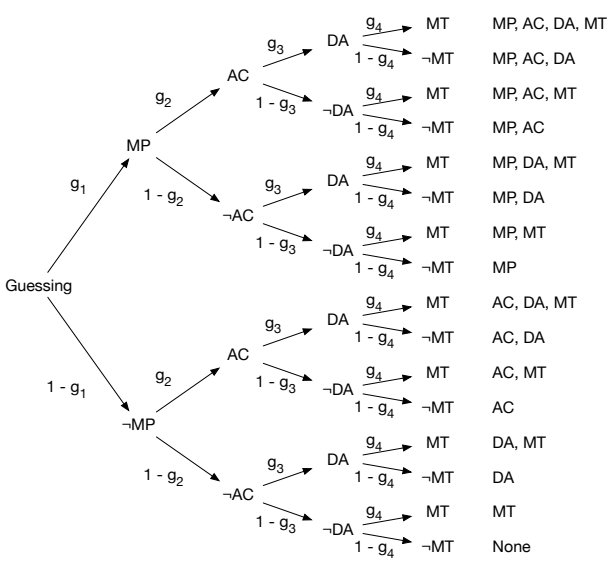


Figure 2: Guessing Subtree. Root node is reached with probability $(1-r)$. Parameters g_1, g_2, g_3, g_4 specify the probability distribution for reaching the 16 possible outcomes. ¬ indicates that the inference is not applied. This type of guessing corresponds to the independence model defined in Klauer et al. (2007).

on a set of three representative strategies taken from recent literature: *Bias Guessing*, *Independence Guessing*, and *Uniform Guessing*.

Bias Guessing. In his original paper, Oberauer (2006) augmented the theoretical accounts of conditional reasoning with guessing subtrees consisting of a single free parameter g representing a bias of accepting any of the four inference types (MP, AC, DA, MT). In this sense, bias guessing represents a basic strategy defined by a single probability parameter representing a reasoner's bias towards applying an inference without relying on reasoning processes. The MPT representing this type of guessing can be obtained from Figure 2 by setting the parameters equal: $g = g_1 = g_2 = g_3 = g_4$. By multiplying the parameters of a certain branch, the corresponding outcome probability is computed. For instance, the pattern (MP, DA, MT) has the probability $P(\text{(MP, DA, MT)}) = g \times (1-g) \times g \times g$.

Independence Guessing. A different way of handling guessing has been used as part of the *Inference-Guessing* model for conditional reasoning (Klauer et al., 2007). Originally devised for the Wason Selection task (Wason, 1968), this model employs a guessing strategy which assigns individual probability parameters (g_1, g_2, g_3, g_4) to each inference. In a sense, it extends on bias guessing by introducing independent biases for each inference increasing its capability to adapt to observed data. The MPT submodel for independence guessing is depicted in Figure 2. Pattern (MP, DA, MT) is assigned the probability $P(\text{(MP, DA, MT)}) = g_1 \times (1-g_2) \times g_3 \times g_4$.

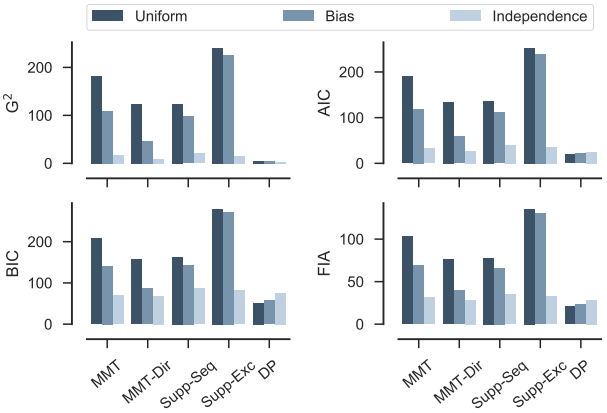


Figure 3: Visualization of the relations between the different metrics.

Uniform Guessing. Uninformed guessing can be represented by a uniform probability distribution over the set of outcomes. In the case of conditional reasoning with 16 different inference patterns producing a binary guessing tree, this corresponds to a fixed g parameter of $g = g_1 = g_2 = g_3 = g_4 = 1/2$. Each pattern is therefore assigned the probability $P(x) = g^4 = 1/16$.

Note, that uniform guessing does not relate processing paths to their corresponding outcomes. Instead it just uniformly assigns probability mass to the set of potential conclusions. This raises the question which properties of guessing differentiate between guessing and alternative reasoning strategies. One possibility to define guessing in a clear distinction from alternative reasoning strategies could be by focusing on context-dependency. If guessing processes are affected directly by the context of the task being modeled, e.g., by the premise information for conditional reasoning, it might be more accurate to refer to them as (heuristic) strategies. Arguably, this notion then ties into the framework of dual-process models (e.g., Evans, 1984).

Results

We analyze the set of models created by combining the conditional reasoning MPTs with the three guessing strategies on the dataset for "basic conditionals" reported by Oberauer (2006). This data was originally obtained by conducting an online study where 343 participants assessed the validity of conditional inferences. Model fits were computed via `MPTinR` (Singmann & Kellen, 2013), a state-of-the-art framework for evaluating MPT models using the `R` environment for statistical computation (R Core Team, 2014). Goodness of fit results as well as the information criteria AIC, BIC, and FIA are reported as produced by the MPTinR analysis.

Table 1: Fitting results of the model-guessing combinations and guessing alone (None).

Cognitive Theory	Guessing	Log Likelihood	G^2	AIC	BIC	FIA	Parameters
None	Uniform	-1902.00	1120.90	1120.90	1120.90	-	0
	Bias	-1856.19	1029.28	1031.28	1035.81	518.82	1
	Independence	-1495.85	308.60	316.60	334.72	168.26	4
MMT	Uniform	-1432.72	182.34	190.34	208.46	103.28	4
	Bias	-1395.82	108.55	118.55	141.20	69.12	5
	Independence	-1349.92	16.75	32.75	69.00	30.97	8
MMT-Dir	Uniform	-1403.68	124.27	134.27	156.92	76.03	5
	Bias	-1364.76	46.43	58.43	85.62	39.79	6
	Independence	-1346.23	9.37	27.37	68.15	28.33	9
Supp.-Sequential	Uniform	-1403.45	123.82	135.82	163.00	77.76	6
	Bias	-1390.26	97.44	111.44	143.15	66.22	7
	Independence	-1351.76	20.42	40.42	85.73	35.04	10
Supp.-Exclusive	Uniform	-1461.70	240.30	252.30	279.49	135.54	6
	Bias	-1454.50	225.90	239.90	271.61	129.96	7
	Independence	-1349.33	15.57	35.57	80.88	32.35	10
Dual Process (DP)	Uniform	-1344.32	5.54	19.54	51.26	20.99	7
	Bias	-1344.32	5.54	21.54	57.79	23.05	8
	Independence	-1343.21	3.33	25.33	75.17	28.34	11

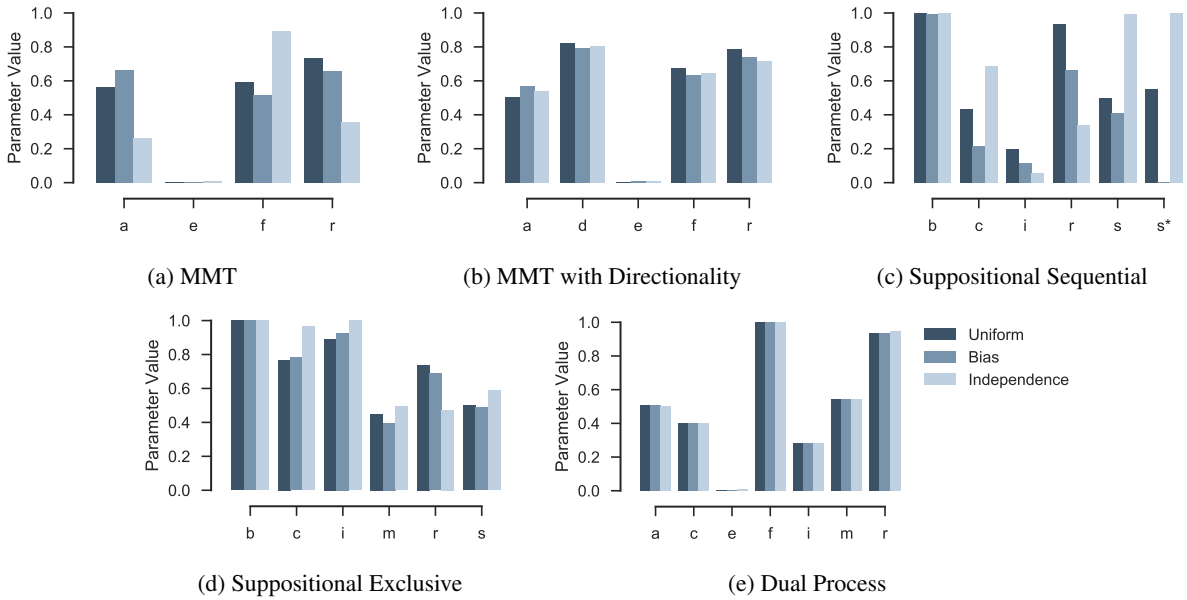
Table 1 and Figure 3 depict the results obtained from fitting the set of models to the data. Apart from the fits of the combined model, Table 1 also contains the results produced by fitting the guessing subtrees alone. Due to the fact that the theories themselves do not account for the complete set of possible outcomes for the conditional reasoning task, the performance metrics without guessing parts could not be determined.

The resulting values illustrate that the choice of guessing has a substantial impact on the overall model performance. Depending on the type of guessing, a wide range of values is obtained. Independence guessing leads to the best performing models, followed by bias guessing, and lastly uniform guessing as the worst option when considering optimality of the fit alone. These results are to be expected as they follow the number of degrees of freedom the guessing strategies add to the model. This is also reflected by the distance between the results of the different types of guessing. Bias guessing, which has only one free parameter is much closer to uniform guessing than independence guessing which features three free parameters. The only exception to this behavior is constituted by the Dual-Process (DP) model, which produces the overall best results and appears to be affected less severely by guessing. However, the comparatively superior goodness of fit results and insignificant variation of the three guessing types suggest that an upper bound of performance is reached. In consequence, the penalty terms of AIC, BIC, and FIA have a bigger impact on the information criteria values.

The guessing models by themselves result in the worst accounts for the data. This is not surprising since guessing does not contain theoretically motivated assumptions about cognition. Instead, these models represent uninformed strategies for producing responses to the task being modeled. Still, when being used as additions to formalized cognitive theories, they are capable of positively influencing the resulting model's performance. By accounting for responses not matching the underlying theory's implications, the predictive power of the theory is enhanced.

Figure 4 illuminates the effects of guessing from the perspective of individual parameter estimates. It shows that the different types of guessing influence the parameter estimates obtained from the fitting procedure. For instance, when considering MMT, the a parameter varies between a value of 0.56 for uniform guessing, 0.66 for bias guessing, and 0.26 for independence guessing. The magnitude of variance observed shows that guessing needs to be applied cautiously when aiming at interpreting the cognitive processes represented by the model parameters. However, the plots also illustrate that the impact guessing has on parameter estimates is dependent on the model itself. MMT with directionality and the dual process model appear to be much less influenced than MMT or the sequential suppositional model, for instance.

Of particular importance is the parameter r which represents the probability of entering the reasoning part of the model instead of relying on guessing for producing a response. Considering the values of r, depicted isolatedly from the other parameters in Figure 5, a considerable influence of guessing can be observed for most of the models. The reason-



(a) MMT (b) MMT with Directionality (c) Suppositional Sequential

(d) Suppositional Exclusive (e) Dual Process

Figure 4: Parameter estimates resulting from fitting the set of models with different guessing trees to the data.

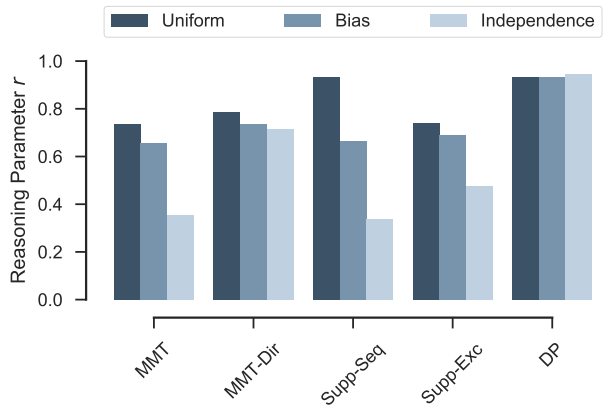


Figure 5: Reasoning parameter r resulting from fitting the theory-guessing model combinations.

ing parameter r differs between 0.73 for uniform and 0.36 for independence guessing. Put differently, by simply switching the type of guessing, the underlying theory is 37% less likely to account for the data. This adds to the observation indicating that the impact of guessing follows the degrees of freedom of the respective strategies. If guessing features larger numbers of free parameters, it is able to account for larger proportions of the data, reducing the importance of the actual reasoning component.

General Discussion

Implementing cognitive theories has become a core aspect of cognitive science. Apart from the raw goodness of fit metrics, interpretability and theoretical merit are essential factors to judge models by. However, when implementing models, the need to add assumptions unwarranted by the underlying theoretical foundation frequently arises. This obfuscates the true power of the theory and may lead to a distortion of resulting qualitative assessments.

Our results show that even seemingly unintrusive additions such as the addition of guessing processes not accounted for by the underlying theory may have unexpectedly high impact on the overall model performance. A shift in the performance of the models following the degrees of freedom available in the guessing trees can be observed. Uniform guessing is not able to be adapted for an optimal fit to the data resulting in the most explanatory weight being assigned to the underlying theory which is reflected by the worst performance values but a relatively high probability of entering the reasoning part of the model. In contrast, bias and independence guessing represent theoretically motivated strategies as alternatives to reasoning. By offering one or three parameters, respectively, for fitting the model, higher levels of performance are achieved at the cost of larger proportions of the data being accounted for by guessing.

When evaluating theoretical accounts on the basis of model implementations, special care needs to be taken to disentangle the original theory's performance from the influence of the additional assumptions. Our results illustrate that there is a fine line between guessing and what must be considered alternative strategies to reasoning. Even when introducing additional processing paths based on a single additional parameter, a hybrid model is formed which produces results that cannot be attributed solely to the underlying theoretical concepts. By disregarding the need for theoretical justification due to treating those alternative strategies simply as "guessing", their potential intricacies, dependencies to the data, and thus influence on the theoretical foundation are obscured.

The results of this work can be generalized to other modeling tasks. Regardless of the framework in use, the addition of alternative processing paths to producing conclusions have an impact on the overall model performance. Without ensuring that the modifications only result in controlled local effects, the soundness of the underlying theoretical assumptions cannot be expected to remain intact. As a conclusion, the role of the model as a representative instance for a theoretical account becomes debatable.

Our findings raise the question as to whether parameterized guessing components can be understood as *guessing* in the first place. Instead, it might be more appropriate to distinguish pre-determined probability distributions with no degrees of freedom as guessing and parameterized versions as (sometimes implicit) realizations of dual-process models with a representation of heuristics. Consequently, guessing might be better defined in terms of context-independent processes that do not depend on the presented information. The present analysis demonstrates the impact of the different types of guessing on reasoning. It highlights the need for a comprehensive theory of guessing.

Acknowledgements

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

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control*, *19*(6), 716–723.

Batchelder, W. H., & Riefer, D. M. (1999). Theoretical and empirical review of multinomial process tree modeling. *Psychonomic Bulletin & Review*, *6*(1), 57–86.

Erdfelder, E., Auer, T.-S., Hilbig, B. E., Aßfalg, A., Moshagen, M., & Nadarevic, L. (2009). Multinomial processing tree models: A review of the literature. *Zeitschrift für Psychologie/Journal of Psychology*, *217*(3), 108–124.

Evans, J. S. B. T. (1984). Heuristic and analytic processes in reasoning. *British Journal of Psychology*, *75*(4), 451.

Evans, J. S. B. T., & Over, D. (2004). *If: Supposition, pragmatics, and dual processes*. Oxford University Press.

Johnson-Laird, P. N. (1990). *Mental models: Towards a cognitive science of language, inference, and consciousness*. Cambridge University Press.

Johnson-Laird, P. N., & Byrne, R. M. (2002). Conditionals: a theory of meaning, pragmatics, and inference. *Psychological review*, *109*(4), 646.

Klauer, K. C., Stahl, C., & Erdfelder, E. (2007). The abstract selection task: New data and an almost comprehensive model. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *33*(4), 680.

Oberauer, K. (2006). Reasoning with conditionals: A test of formal models of four theories. *Cognitive Psychology*, *53*(3), 238–283.

Purdy, B. P., & Batchelder, W. H. (2009). A context-free language for binary multinomial processing tree models. *Journal of Mathematical Psychology*, *53*(6), 547–561.

R Core Team. (2014). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `http://www.R-project.org/`

Ragni, M., & Tse, A. P. P. (2017). Cognitive Computational Models for Conditional Reasoning. In M. K. van Vugt, A. Banks, & W. Kennedy (Eds.), *Proceedings of the 15th International Conference on Cognitive Modeling* (p. 109-114). Coventry, United Kingdom: University of Warwick.

Riefer, D. M., & Batchelder, W. H. (1988). Multinomial modeling and the measurement of cognitive processes. *Psychological Review*, *95*(3), 318.

Rissanen, J. J. (1996). Fisher information and stochastic complexity. *IEEE transactions on information theory*, *42*(1), 40–47.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, *6*(2), 461–464.

Singmann, H., & Kellen, D. (2013). MPTinR: Analysis of multinomial processing tree models in R. *Behavior Research Methods*, *45*(2), 560–575.

Verschueren, N., Schaeken, W., & d'Ydewalle, G. (2005). A dual-process specification of causal conditional reasoning. *Thinking & Reasoning*, *11*(3), 239–278.

Wason, P. C. (1968). Reasoning about a rule. *The Quarterly journal of experimental psychology*, *20*(3), 273–281.

ACT-Droid Meets ACT-Touch: Modelling Differences in Swiping Behavior with Real Apps

Nele Russwinkel (nele.russwinkel@tu-berlin)
Sabine Prezenski (sabine.prezenski@tu-berlin.de)
Lisa Dörr (lisa-madeleine.m.doerr@campus.tu-berlin.de)
Department of Cognitive Modeling in dynamic Human-Machine Systems, TU Berlin,
Marchstr. 23, 10587 Berlin, Germany

Frank Tamborello (frank.tamborello@cogscent.com)
Cogscent, LLC, Houston, Texas, USA

Abstract

This paper presents a tool (ACT-Droid) that integrates user models with mobile devices and enables modelling time-sensitive touch interactions via ACT-Touch. It demonstrates that using ACT-Droid in combination with ACT-Touch is a step towards model-based user studies of real smartphone apps. A special focus is set on modeling swipe interactions. Hereby, different interaction strategies depending on preknowledge of the interface type (e.g. sorted or unsorted lists) are modeled.
Results of an empirical study conducted with a real smartphone app indicate that searching through lists results in different kinds of swipe behavior for sorted vs. unsorted lists. Sorted list interfaces elicited fewer and quicker swipes than unsorted lists.
This paper introduces ACT-R model approaches capable of capturing such behavior.
Finally, the importance of understanding top-down strategies involved in such regular tasks is discussed.

Keywords: Applied research, ACT-R, User Modelling, item search, swipes, Touch interaction, tool.

Applied Cognitive Modelling

To test the validity of theoretical approaches in real word scenarios, applied tasks are needed. The question hereby is, whether developed theoretical concepts also apply for realistic, less-controlled tasks. Already experiments outside the modeling community are conducted on mobile devices (e.g. tablets, smartphones) or on online platforms (e.g. mechanical Turk), since this is often more convenient. Modelling such tasks often neglects the kind of interaction.

Not only different devices, but also different interaction modalities (such as swiping, steering, tapping) not only influence how we interact with devices but also how we plan and represent tasks internally and how we judge an interaction as successful, since the feedback could be different.

Another interesting question is: How do we adapt our behavior with specific devices we use every day? Over time we develop strategies to be more efficient in our tasks. These strategies help users to achieve easier and more effortless interactions. In this manner, interfaces that allow such effortless interaction will be judged as being of high usability. More so, if users find efficient strategies for device interaction this changes the perceived usability of a device. For example, Burnett et al (2013) showed that the speed and length of touch screen interaction differs if users are preoccupied with another task (e.g. driving) then when they are focusing on just interacting with a touchscreen. The authors speculate that observed slower swiping behavior during driving is due to reduced resources available in such task. A cognitive modeling approach has the potential to shed light on such claims.

Gray & Boehm-Davis (2000) identified that people are sensitive to the costs of low-level processes in their interactive task behavior and adapt their behavior accordantly. Especially for reoccurring strategies in low level steps of interactive behavior, these strategies quickly add up and produce relevant time differences that user models should be able to account for.

However, to address these topics appropriate tools and approaches are needed. These should shed light on understanding what factors, strategies and task demands influence interaction with mobile devices. Cognitive Modelling is a relevant method for understanding interaction behavior (Ritter, Van Rooy & Amant, 2002). Cognitive modeling can help to develop theoretical concepts and allows to predict behavior for new tasks and devices. This can be used as a validation for the theoretical approaches. To support research in this area we will present cognitive modeling approaches for ACT-R (Anderson, 2007)

In this paper we present the newest version of ACT-Droid (Dörr, Prezenski & Russwinkel, 2016), a tool that allows a cognitive model to connect with an application on a mobile device. ACT-Droid can be used with any open-source Android application. It allows ACT-R models to directly interact with the interface of a mobile device. A new feature is that ACT-Touch commands are now integrated in ACT-Droid. Thus, directly modeling long tap, touch, swipe and other kinds of interactions on the actual Android application is now possible.

We want to show that our modeling approach allows the capture of user behavior occurring in natural tasks such as efficiently finding and selecting items. Therefore, the differences in search time, and behavior evolving due to the use of different strategies are examined in an empirical study. A modeling approach which integrates these different swipe interactions is presented.

ACT-Touch Introduction

ACT-R includes a library of HCI routines. However, these routines assume a traditional desktop computing environment. Hereby, the modelled user views an upright, freestanding computer monitor while holding hands upon a keyboard's home row and occasionally reaching for a one-button mouse (Bothell, 2017). However, since the iPhone popularized the handheld, multi-touchscreen mobile computer starting in 2007, Apple has produced 1.2 billion iPhones (Washington Post, 2017) and Android has more than two billion monthly active users (Statista, 2017). Mobile computing has eclipsed desktop computing (Ofcom, 2017, p. 147) and HCI modeling tools must address this reality.

ACT-Touch (Tamborello, 2018) is an add-on to the ACT-R software distribution. It provides a library of additional motor commands for the model to execute in the milieu of a simulated multi-touch display computing environment. For example, contrasting with canonical ACT-R's default hand position, ACT-Touch assumes both hands are positioned relative to the display, rather than the keyboard, on a coordinate system corresponding to the pixels of the display, floating approximately 1 inch above the display surface. ACT-Touch enables ACT-R's hands to move around the display and to perform many of the gestures familiar to users of mobile multi-touchscreen devices, such as taps, swipes, and pinches. While some of these motor commands, such as tap, are essentially repackaged extant ACT-R commands renamed to reflect the multi-touchscreen environment, other commands, such as swipe, depart further from ACT-R's desktop environment.

ACT-R movements use features, such as hand, finger, distance, and direction, to describe movement. Models compose these features into a movement and ACT-R calculates movement preparation, execution, and finish time from those features. Tap is peck renamed, including hand, finger, distance, and direction. Swipe, on the other hand, includes two additional features: number of fingers and speed. Number of fingers is important to the swipe gesture because some touch interfaces, such as iOS on the iPad, interpret the varying numbers of fingers in the swipe gesture as unique commands, to, for example, scroll a document or switch to iOS' multitasking display (Patterson, 2013). The project reported herein implemented a new speed feature for the swipe motor command to allow models to control swipe speed. This is important because, e.g. iOS scrolls more of a document with each pixel of the display traversed by the fingers the faster the fingers move, accelerating scrolling behavior in response to accelerated user input.

ACT-Droid Tool

In the future it may be possible to use cognitive models instead of user testing (see Prezenski & Russwinkel, 2014). However, studies with human participants are necessary for most user-study related questions and thus, also real human-computer interfaces. In a smartphone setting these are either Android or iOS applications. Thus, until recently an ACT-R modeler interested in modelling user-behavior had to translate the entire GUI into Lisp.

ACT-R‘s GUI, the AGI, is limited in several ways. For example, there are only three different types of objects that can be created for interactions: text, line and button. These objects have limitations themselves, as one cannot create a button with background color and font color.

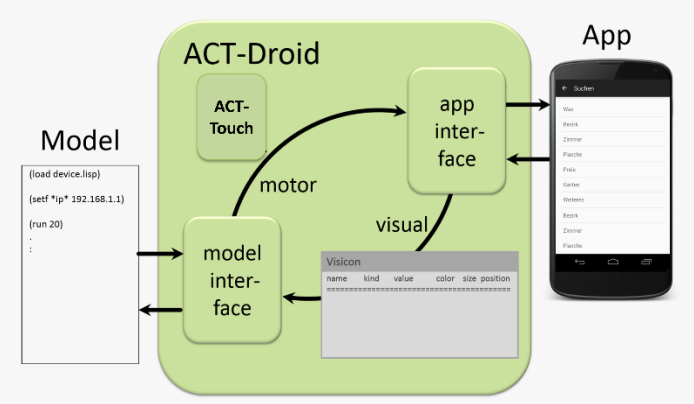


Figure 1: Structure of ACT-Droid

To avoid such restrictions and to realize direct interactions between ACT-R models and applications we developed the tool, ACT-Droid. This tool connects an ACT-R model with an Android app, thus enabling the model to execute its motor commands on the app and the app to fill ACT-R‘s visicon with the visible objects it displays. See Dörr, Prezenski and Russwinkel for more details on how ACT-Droid works. The first version of ACT-Droid could not cope with touch interactions.

Now, ACT-Droid performs the motor output via ACT-Touch commands (e.g. swiping) on the app and subsequently updates the visicon (if the app screen changes). These functionalities are provided by the model interface and the app interface, which communicate with each other (see figure 1).

For our study an example real-estate app was installed on a smartphone. ACT-R communicates with the app over TCP/IP sockets. Hereby, the app interface establishes a server socket and the model interface connects ACT-R as a client.

ACT-Touch commands are used for motor output. So, each time the model’s finger is moved, the model interface sends the new position to the app interface. The app interface saves the current position of the finger. Furthermore, if the command to tap, swipe etc. is received, the app interface performs it at the saved cursor position.

The app interface provides already a basic description of all visible information and it can be further adjusted to different apps. Hence, higher fidelity modeling of user behavior within a mobile context should be possible with ACT-Droid. Thus, it is now possible to write ACT-R models that capture slow or quick swipes occurring while using a real app.

In this paper a study is reported that investigates different scrolling behavior. This data is used to develop a modeling approach. This study can be regarded as a proof of concept to show the usefulness of this tool to test theories in applied settings. Since the focus of the paper is set on the tool only the most relevant data is presented.

Modelling Approach for Swipe Interactions

Swipe Mechanics

First, the Lisp environment loads ACT-R and then ACT-Touch. After this the motor commands ACT-Touch provides are available to any ACT-R model just as canonical ACT-R motor commands are. Within the manual request movement type other request slot values indicating the movement features are defined. For example, to swipe with the right index finger rightward with two fingers for 100 pixels at a moderate speed, the production's righthand side should indicate:

```
+manual>
cmd swipe
hand right
finger index
r 100
theta 0
num-fngrs 2
speed 3
```

ACT-R uses Fitts' Law to compute movement execution times. ACT-Touch's swipe execution time function calls ACT-R's Fitts' law function with coefficient of difficulty inversely squared to the speed feature the model supplies to the swipe command. These are execution time, as well as a width parameter set to the width of the display, 500 pixels in the case of the example model. See the compute-exec-time method specified for the swipe class in act-touch.lisp, where it calls ACT-R's fits function with the motor module.

Model: Swipe in sorted and unsorted lists

In the following, we will introduce a simple, errorless performance model that uses preknowledge of whether a long list of items is sorted alphabetically or randomly to decide its interaction strategy. The model will either swipe with a fast speed to scroll the interface quickly to the end, where it "thinks" its target is positioned, or it will swipe slowly to scroll the interface slowly as it exhaustively searches the list for its target.

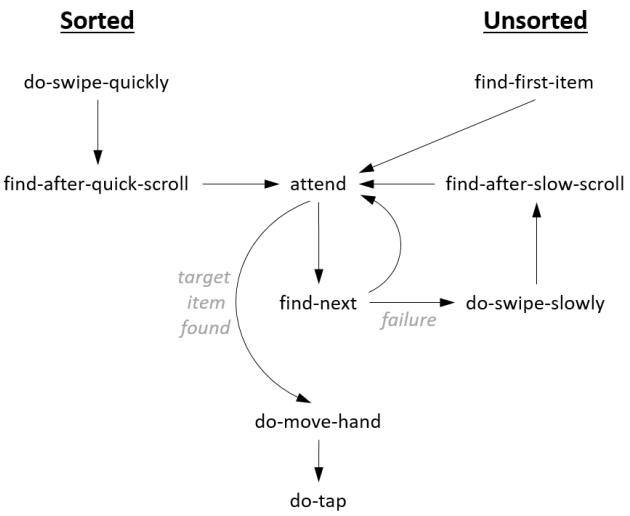


Figure 2: Production graph

If the of the model indicates that the list is ordered, the model will start the search task by performing a quick, short swipe to get to the end of the list (do-swipe-quickly) (see figure 2). Afterwards, a search through the visible items from top to bottom (find-after-quick-swipe, attend, find-next, attend, …) is performed. If the chunk in the visual buffer matches the correct item, it moves the hand (do-move-hand) and taps on the item (do-tap). This takes the model 3.008 seconds.

If the list is randomly ordered (preknowledge indicates a random list), then the model must search this list exhaustively. The model will omit the quick swipe in the beginning and start searching from the top (find-first-item, attend, find-next, attend, …). When no visual-location below the current can be found, the model "knows" that it reached the bottom of the screen and swipes slowly to make about four new items visible (do-swipe-slowly). This is repeated until it finds the correct item. It takes 4.112 seconds to find and select the target item.

ACT-Droid including this swipe model example can be downloaded from http://dx.doi.org/10.14279/depositonce-5181. Detailed instructions on how to install and use ACT-Droid can be found in "Readme.txt".

Study Searching through Lists via Scrolling

To demonstrate that different cognitive strategies are used we looked at targets positioned at the same place on a searchable list (e.g. upper position vs. lower position in a list over several screens) that result in different search times and qualitative behaviors depending on the organization of the lists. Furthermore, we want to show that an ACT-R model connected with ACT-Droid and using ACT-Touch can resemble these behavioral patterns, both quantitatively and qualitatively.

More precisely, **different** scrolling behaviors should emerge depending on whether the lists are ordered randomly. On the one hand, the participants are required to inspect each item in the random lists visually. It is expected that a random

order will result in slower scrolling behavior. On the other hand, with the ordered version (either via alphabet or numbers) participants can use their expectations to guide their search behavior. Thus, it is expected that they scroll faster (or flick) until they reach the predicted position of the target.

A further proof of concept was to test to what extent participants touch behavior such as scrolling and regular behavior can be captured with ACT-Touch in combination with ACT-Droid.

Application: The study was conducted with a modified version of a real-estate application for Android (Prezenski & Russwinkel, 2016). This application has a hierarchical list style design. Different search criteria for real-estate, such as city district, number of rooms, or additional features, can be selected to define a search for real-estate. The criteria are presented in hierarchically organized lists. Each list can range over 1 to 3 screens. For example, to a select 3-room-apartment, users are required first to select room-size and then select "3" in a list.

In the study, two versions of the application were used. In one version the criteria in the lists are ordered either alphabetically or numerically (ORDER) in the other version they are ordered randomly within the lists (RANDOM).

Task: The participants were asked to imagine that they were searching for real-estate together with a relative of theirs who is not familiar with smartphone usage. The task of the participants was to search for different properties. Hereby, they had to search and select criteria (e.g. "apartment" with "3 rooms" and "70 m²" in "Berlin" in the district "center" with a "balcony" for "800 €" and an "elevator") using the app. The criteria were read to the participants one by one. The criteria required the participants to scroll down in the app. The app was installed on a Google Nexus 4 smartphone (4.7-*inch*, 1280 x 768 *pixels, 320 ppi)*, running Android 4.4 (Kitkat).

The participants were instructed to hold their smartphone in their hand. This was done to achieve a more natural interaction posture.

Design: For each real-estate search, eight different criteria had to be selected. The participants searched for real-estate with varying criteria three times. After this, the version was changed without notifying the participants, and they searched three times again for different real-estate with eight criteria. Half of the participants began with the ORDERED version, the other half with RANDOM version.

To ensure that differences in search time resulted from the scrolling behavior and not on the positioning of the criteria, the target criteria of both versions were positioned at the same places in both versions of the app.

The "task time" was measured from the selection of the correct criterion to the selection of the correct item. Or in other words, the time the participants needed to find an item in a list. Only the swipes during that "task time" and longer than 100 pixels were considered. Touch interactions for a shorter distance were assumed to be taps, long taps, or a gesture that had no consequences for this experiment. The "speed" of the swipes is provided in px/ms with a screen resolution of 320 ppi. As a straightforward analysis we calculated the mean speed of every swipe during a search within a criterion and then took the average over them. This way we had an indication of how fast/slow the participants scrolled.

Participants: The participants were 30 students (17 female and 13 male), between 21 and 31 years old. 28 were right-handed and 2 were left-handed. Their average self-reported smartphone usage was 3.46 hours per day. They self-rated their experience with smartphone apps 3.47 (sd. 1.25) on a scale from 1 to 5 (1 being no experience and five being a lot of experience).

Results

Trials with small errors like swiping too far were not discarded by default. But outliers were handled systematically, data points of more than 2.5 interquartile distance from mean value were discarded. The calculated t-tests focused on general differences between interaction behavior for sorted and unsorted lists (compare with figure 3). Significant differences were found for tasktime, touches and speed. Paired (tasktime) $t(29) = -9.825$, $p \leq 0.001$; paired (touches) $t(29)=-6.125$, $p \leq 0.001$; paired (speed) $t(29)=6.275$, $p \leq 0.001$.

Thus, for a better analysis we divided items according to upper (beginning of list – first page), middle and lower items which were located at the end of the list. It was expected that the difference for sorted and unsorted lists was especially pronounced for items at the end of the list – we called lower items (see figure 4), that incorporated probably more touch interactions. Therefore, a special analysis for those cases was made. Again, a t-test analysis was conducted.

Significant differences for lower items were found for tasktime, touches and speed between sorted and unsorted lists. Paired(tasktime_low) $t(29)=-9.393$, $p \leq 0.001$; paired(touches_low) $t(29)=-4.812$, $p \leq 0.001$; paired(speed_low) $t(29)=6.241$, $p \leq 0.001$.

The difference in task time between sorted and unsorted lists might also account for the visual search behavior and not just swipe behavior. Participants needed more visual attention to inspect numerous items especially for low positioned targets, because the target could have been anywhere. In the ordered search, participants needed this search behavior only for the items at the end of the list. The speed and number of touches in the different conditions give us a better impression of the different touch interaction user strategies, as explained above.

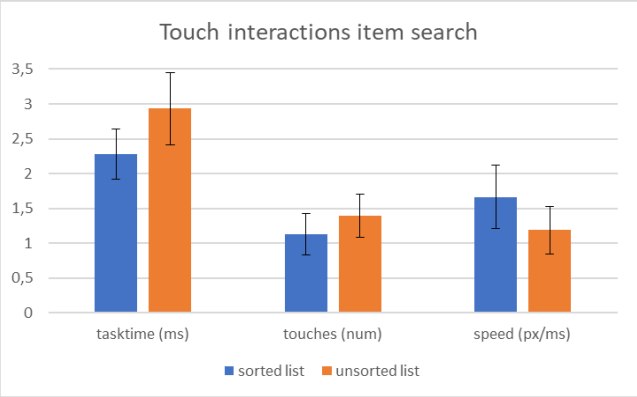


Figure 3: Touch interactions divided for sorted and unsorted lists (including standard deviation)

The results show more touches for unsorted lists. The number of touches would have been more pronounced for even longer lists than we tested. The results also show higher speed in swipes for sorted lists, probably used to quickly get to the part of the list where the target is expected to be.

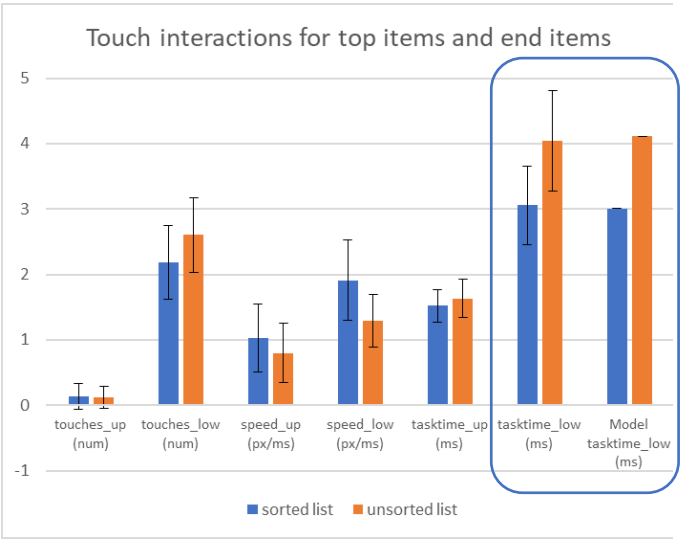


Figure 4: Touch interactions divided for lower and upper search targets of participants (incl. std.), and comparison to Model data for task time_low

Those time differences and touch differences do matter because they accumulate for every search list and already show a difference of one second for one selection step in our relatively short lists. These data represent what differences arise in such reoccurring general task components and that it is necessary to account for such differences in the task model in cases where reaction times are relevant.

A more detailed analysis of the specific kind of different touch interaction is the topic of another publication. For being able to capture interaction times for applied tasks and evaluation of technical systems we do not need to model the touch interaction in detail. It is sufficient to capture the general differences if we know that we find the item at the end of the list or not. This is not necessarily always caused by a sorted or unsorted list but can be the result of learning certain visual location or navigational heuristics. Having a good idea where and how to find the target quickly is what makes the difference.

Discussion

We introduced a new tool, which connects an Android app with an ACT-R user model. Furthermore, the tool makes prototyping of the application in Lisp obsolete by allowing the model to directly act upon the real app. This tool uses the ACT-Touch commands to simulate realistic touch interactions thus making it possible to evaluate user interfaces and interactive behavior more precisely and provide a deeper understanding of the underlying processes.

In the current paper, empirical data is presented to show how different kinds of swipe interactions are used and what impact this has on task time. These behavioral differences may not only occur due to sorted and unsorted lists. Other reasons for different swipe behavior might be preknowledge about the position of certain icons or targets or strict time constraints. More research is needed to understand these kinds of influences on behavior.

Realizing scrolling behavior has been a major burden in several mobile device experimental setups. Scrolling behavior was a necessary touch interaction in most applications that we looked at. Furthermore, task time is a relevant key factor for usability tests evaluating whether one application is better than competing apps. Our presented approach solves these problems.

Since more and more touch displays are integrated into larger technical systems such as cars or the cockpit of airplanes, we find a pronounced request from industry to understand how operators work with such touch displays within a realistic task.

Now that scrolling behavior on touch displays can be modeled in simulation, this could be helpful for developing and evaluating assistant systems, especially in driving scenarios (e.g. Salvucci, 2006). These models would be tools to understand, describe and predict in detail how operators interact with technical systems. This method could help to decide which kind of interface design is more effective in some sense than another.

References

Anderson, J.R., Bothell, D., Byrne, M.D., Douglas, S.,Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111, 1036-106. al Review*, 111, 1036-106.

Bothell, D. 2017. ACT-R Reference Manual. http://act-r.psy.cmu.edu.

Burnett, G., Crundall, E., Large, D., Lawson, G.,& Skrypchuk, L. (2013). A Study of Unidirectional Swipe Gestures on In-Vehicle Touch Screens. In Proceedings of the 5th International Conference on Automotive User

Interfaces and Interactive Vehicular Applications (pp. 22-29). ACM.

Doerr, L.-M., Russwinkel, N., & Prezenski, S. (2016). ACT-Droid: ACT-R Interacting with Android Applications. In D. Reitter & F. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (pp. 225-227). University Park, PA: Penn State.

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), 322-335.

Halbrügge, M. and Russwinkel, N. (2016). The sum of two models: how a composite model explains unexpected user behavior in a dual-task scenario. In D. Reitter & F. E. Ritter (Eds.), *Proceedings of the 14th International Conference on Cognitive Modeling*. University Park, PA: Penn State.

Ofcom. 2017. International Communications Market Report 2017. https://www.ofcom.org.uk/__data/assets/pdf_file/0032/108896/icmr-2017.pdf

Patterson, B. 2013. iPad tip: 3 nifty iPad gstures you need to try. https://heresthethingblog.com/2013/12/11/ipad-tip-3-nifty-ipad-gestures/

Prezenski, S.& Russwinkel, N. (2014). Combining cognitive ACT-R models with usability testing reveals users mental model while shopping with a smartphone application. Int. J. Adv. Intell. Syst., 7(3), 700- 715.

Prezenski, S. & Russwinkel, N. (2016). Towards a general model of repeated app usage. In D. Reitter & F. E. Ritter (Eds.), Proceedings of the 14th International Conference on Cognitive Modeling (pp. 201- 207). University Park, PA: Penn State.

Ritter, F. E., Van Rooy, D., & Amant, R. S. (2002). A user modeling design tool based on a cognitive architecture for comparing interfaces. In Computer-Aided Design of User Interfaces III (pp. 111-118). Springer, Dordrecht.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. Human factors, 48(2), 362-380.

Statista. 2017. "Number of Google Play Store apps 2017 | Statistic". Retrieved 2018-01-03.

Tamborello, F. 2018. ACT-Touch. https://github.com/tamborello/ACT-Touch/

Tsukayama, H. 2017. "Apple stock soars to a record high on great earnings and a strong forecast for the next iPhone". The Washington Post. Retrieved August 2, 2017.

An Architecture Approach to Modeling the Remote Associates Test

Jule Schatz (schatzju@umich.edu)
Steven J. Jones (scijones@umich.edu)
John E. Laird (laird@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

Abstract

The remote associates test (RAT) depends heavily on memory retrieval and is difficult for humans. A previous model of difficulty on this task accounted for difficulty with a measure incorporating fan and association strength. This paper investigates how the choice of knowledge base and agent strategy impact difficulty on the task while providing a more comprehensive account for human difficulty on this task in terms of cognitive architecture components. The models we created, using the cognitive architecture Soar, vary by using two distinct methods of retrieval from semantic memory. The knowledge bases used in our models vary in that one uses only collocations and compound words to form word associations while the other is from a crowd-sourced dataset with unrestricted types of word association. The model which best matches human difficulty relies on spreading activation to drive retrieval and uses the crowd-sourced dataset for its knowledge base.

Keywords: Semantic Memory; Remote Associates Test; Soar; Association-based Retrieval.

Introduction

This paper investigates computational models for the remote associates test (RAT) (Mednick, 1962). A single RAT problem consists of presenting three words and then asking the test taker to respond with a forth word that is associated with the three given words. For example, if "Swiss," "cake," and "cottage" are the given words, then "cheese" would be the correct response. Bowden and Jung-Beeman developed 144 RAT problems and tested human performance on those problems. To minimize any variance from confounding factors, they used only compound word or phrase associations. For example, the association between "deep" and "sleep" is valid because those two words are often found next to each other. The association between "deep" and "complex" is not valid because "deep" and "complex" do not form a compound word or common phrase even though they are associated through similar meanings. The problems use common words, to avoid vocabulary difficulties. To avoid priming effects, solution words are never repeated or used as problem words. Three additional example problems are shown in Table 1. The human study included four time limits: 2 seconds, 7 seconds, 15 seconds, and 30 seconds. The paper provides the mean time to solution, the standard deviation for time to solution, and the percent of participants that correctly answered each question in each time limit.

Word 1	Word 2	Word 3	Answer
man	glue	star	super
dew	comb	bee	honey
rain	test	stomach	acid

Table 1: Example RAT items. The answer is associated with the words through collocation or as a compound word.

As opposed to generally characterizing word association memory (Griffiths & Steyvers, 2002), Olteţeanu and Falomir (2015) created a model intended to provide an account for human performance on those 144 RAT problems. Their model, comRAT-C, was created within the CreaCogs architecture (Olteţeanu, 2014) and was inspired by their account of creative problem solving which posits two extremes of behavior: "creative search" and "productive representation construction processes". The "creative search" extreme is embodied in comRAT-C, which uses associational links to search a knowledge base for a representation that affords a solution to a problem. Their knowledge base, called RAT-KB, builds off of the most frequent 2-grams from the Corpus of Contemporary American English (COCA) (Davies, 2008). Associational links in RAT-KB are bidirectional.

In their analysis (Olteţeanu & Schultheis, 2017), they state that the difficulty of this task depends on "(i) the frequency of a query-answer association, as a form of associative strength and (ii) the ratio between such an associative strength and the number of answer associations." We interpret these factors as being analogous to how association strength (Anderson & Pirolli, 1984) and fan (Anderson, 1974) govern retrieval difficulty. Their results were not in terms of providing a model with matching timing and correctness. Instead, they show the correlation between their difficulty estimates for RAT items and human data, i.e., both human timing data and human correctness data on the RAT items. We interpret this as characterizing the relative difficulty for humans by the ordering of human solution time and human correctness.

We use their work as inspiration for our research but deviate in hopes of providing a more comprehensive analysis. First, we note that RAT-KB includes only associational links for collocations and compound words, and that all links are bidirectional. Humans know many words and associations beyond this constraint, and possibly do not have bidirectional links between these words. Thus, our first step is to use a larger, more comprehensive knowledge base, where links are not necessarily bidirectional. We then determine how such a knowledge base influences task performance, and more specifically, relative problem difficulty. Second, we wish to determine whether existing architectural long-term declarative memory retrieval theories, as developed in ACT-R (Anderson, 2009), are sufficient to accurately model RAT problem difficulty. In these declarative memory models, retrieval is determined by base-level activation, association strength

(Anderson & Pirolli, 1984), and fan (Anderson, 1974). To explore these questions, we develop models in Soar (Laird, 2012) whose long-term declarative memory retrieval mechanisms mimic those in ACT-R (Jones, Wandzel, & Laird, 2016). Third, there does not exist a simple, deliberate model of retrieval that does not primarily rely on association strength or fan that can be used as a baseline for comparison with the association-based models. Thus, our third step is to develop such a model in Soar which uses queries that are not influenced by associate strength or fan except in the case of ties.

In the remainder of the paper, we proceed through these steps, one by one. First, we introduce a new crowd-sourced knowledge base. Second, we describe the two models we developed in Soar. Third, we evaluate these models on the new knowledge base as well as on a replica of the original RAT-KB knowledge base, focusing on how well these models (and knowledge bases) model human difficulty. The primary result is that the more comprehensive knowledge base combined with associational retrieval allow us to model human difficulty with high correlation ($R^2 = 0.89$).

Knowledge Bases

To allow us to compare our new knowledge base to prior work, we reconstructed RAT-KB, by creating a knowledge base called COCA-TG based on the steps described in the original paper. The final number of words and associations for COCA-TG are shown in Table 2. RAT-KB includes bidirectional associations for all words, as does COCA-TG.

Because COCA-TG leaves out other types of associations that can indirectly influence retrieval (e.g. through competition and the fan effect), we created a larger knowledge base using the Human Brain Cloud (HBC) database. HBC was crowd-sourced through an online game of word associations (Gabler, 2013), where the player is presented with a word and asked to type in any other word that they believe to be closely related to the given word (if the given word is "bird" the player might type "feather" or "fly" or "nest"). The website records the human responses, and creates a dataset that consists of triples in the form of "word1," "word2," and weight, where weight is the number of times "word1" was associated with "word2". HBC only includes links entered by a player, so not all word pairs have bidirectional links (as in COCA-TG and RAT-KB). As shown in Table 2, HBC contains over ten times as many words, and over six times as many associations as contained in COCA-TG.

	HBC	**COCA-TG**
words	231,851	20,809
associations	2,403,203	349,196

Table 2: For each knowledge base, the number of unique words, and associations between words. Bidirectional associations count as two associations.

Models

Our models are developed in Soar, which features a long-term semantic memory that can be queried to retrieve information into working memory (Derbinsky & Laird, 2010). To run a model, semantic memory is initialized with the contents of a knowledge base, where nodes in the memory consist of words and the links are associations. The weights of the associations are those in the knowledge bases.

Retrieval in Soar returns the most highly-activated element which satisfies the provided cue. The activation of an item is the sum of base-level activation and spreading activation. Base-level activation represents the frequency and recency of prior retrievals, but for these models we had no prior values. Instead, we initialized all words in the knowledge base with a single base-level activation. However, we assume that words should have some usage history and we return to this issue in the discussion.

To solve a RAT problem, a model uses the three presented words to find the associated answer. There are potentially many strategies for doing this; however, we focused on two strategies that are directly supported in Soar. In Soar, an agent can retrieve information from semantic memory, either by providing a specific cue that is matched against elements in long-term memory (Cued Retrieval model), or an agent can use a general cue and leverage spreading activation to retrieve words based on context as defined by the contents of working memory (Free Recall model). The Cued Retrieval model uses queries that include the original words, whereas the Free Recall model does not include the original words and relies on spreading activation, which incorporates both association strength and fan.

Cued Retrieval

As a baseline, we created the Cued Retrieval model, with the goal of correctly answering as many RAT items as possible given the knowledge available in long-term memory, while keeping agent design simple and in accordance with architectural constraints. This model first retrieves all three given words into working memory. It then creates a cue that specifies semantic memory should only return a word that has an outgoing link to all three of the given words. Semantic memory then returns either a word that matches the cue (that is associated to all given words), or it reports a failure if no such word exists. If semantic memory has multiple possible solutions, spreading activation acts as a tie breaker. If the initial query failed, the model changes the cue to only require semantic memory to return a word that is associated to two of the given words. The model will try all combinations of two words, and report an answer for the first one it finds. If all of those fail, it tries each given word individually and reports that answer. Because the model deliberately queries semantic memory for a word with all associations first, it will find a correct solution if one exists in the database, which is not guaranteed in the Free Recall model.

Free Recall

The Free Recall model incorporates association strength and fan via spreading activation (Jones et al., 2016). The agent first retrieves the three given words into its working memory from semantic memory. Having these words in working memory causes activation to spread to words linked to those words in semantic memory. Each given word acts as a source for an equal amount of activation, which is then divided proportionally among the outgoing links based on association strength. Association strengths from a given source are normalized to sum to one. Activation decays with the distance of spread, but in this model, for simplicity, spreading is limited to a depth of 1.

Consider an example where there is a source word s and recipient word r with a pre-normalized association strength from s to r of $a_{s\to r}$. Assume a set, R, of all recipients. The contribution of spread from the source to the recipient in this case is $\frac{a_{s\to r}}{\sum_{k\in R} a_{s\to k}}$. Therefore, an item with a stronger association from the source word will get more activation than one with a weaker association. In addition, the more links or fan the source has, the less activation will spread to its recipients.

To retrieve a word, the model initiates a retrieval from semantic memory with the only constraint being that the word is not one of the three given words. Semantic memory then returns the word with the highest activation. A high activation is no guarantee that the retrieved word is associated with all three words because words can be retrieved that have strong associations to only one or two of the original words, especially if they have low fan. Once a word is retrieved from semantic memory, the model tests how many of the three initial words relate to it by testing if there are links between it and those initial words. If it is related to all three words, the model uses the word as its solution. If the retrieved word is related to two or fewer of the given words, the model queries again and retrieves a new word from semantic memory, inhibiting any it has previously retrieved. The number of attempts it will make is a parameter, which we vary in the evaluation. If the model runs out of attempts, it chooses one of the retrieved words that has the most relations with the given words.

This model incorporates the findings from Olteţeanu and Schultheis's research in terms of the two factors (association strength and fan) that influence the difficulty of RAT items for humans. Their findings indicated that those factors influence whether humans can solve a RAT problem and the time it takes for them to find a solution. Words that have stronger associations are more likely to be retrieved by this model, as well as words from low fan sources.

Evaluation

We tested both the Cued Retrieval and the Free Recall models using both the HBC and the COCA-TG databases, giving four model configurations. Our results compare the models' timing and correctness to human timing and human correctness on the task, focusing on correctness. ComRAT-C provided a probability value that they consider an estimate of the probability that a word is an answer. Their results were that for a RAT item with a given correct answer, comRAT-C's estimate of the probability that the correct answer was correct correlates positively with the number of humans who answered correctly and correlated negatively with human mean time to solution. However, Soar models retrieval as competitive, so only a single element is selected. The significance of this is that the activation of the correct answer does not completely determine if it will be the model's answer. An activation can be high, but if it is not the *highest* with respect to words that compete for retrieval, then it will not be retrieved. For this reason, our results are not directly comparable with those of comRAT-C. We instead adopt an approach where we compare the correctness of the answers produced by our models to the correctness of the human answers.

Overall Difficulty

First, we consider overall task difficulty as it relates to modeling difficulty on the RAT. Figure 1 shows two diagrams, one for each knowledge base. These diagrams include the number of RAT items that were answered correctly for different model configurations, as well as the average number of correct responses made by humans. These averages are for when humans have only 15 seconds and 7 seconds to generate an answer, and as is obvious, this is a difficult task for humans. The x-axis is the number of attempts (1-20) for the Free Recall model. The models for humans (light and dark green) and Cued Retrieval (dots) have only a single attempt. We show them as straight lines for ease of comparison with the Free Recall model. We also include the number of items where all the given words and the answer exist in the database.

The top figure shows results from using COCA-TG for both models. The Cued Retrieval model with COCA-TG gets 65 RAT items correct. The Free Recall model initially improves as more attempts are made, and achieves better performance than the Cued Retrieval model from 3 attempts on. The best it achieves is 78 correct at ten, eleven, and fourteen attempts. This improvement is possible because this model makes guesses for problems where it cannot find an exact answer, and sometimes those guesses are correct. As the figure shows, the COCA-TG models outperform humans except in the case where the Free Recall model makes a single attempt.

The bottom figure shows results using HBC. HBC contains more correct answers than COCA-TG (105 vs. 55), invariably because of its larger size (see Table 2). Once again, through guessing, the Free Recall model achieves performance better than one might expect. With HBC, Free Recall achieves the 7 seconds human performance when it uses two attempts and the 15 seconds human performance with three attempts. We hypothesize that more attempts are required to achieve the same performance in the HBC database, because, on average, HBC words have higher fan (32 vs. 17).

Relative Difficulty

Next, we compare the results to human data provided by Bowden and Jung-Beeman. We are interested in which

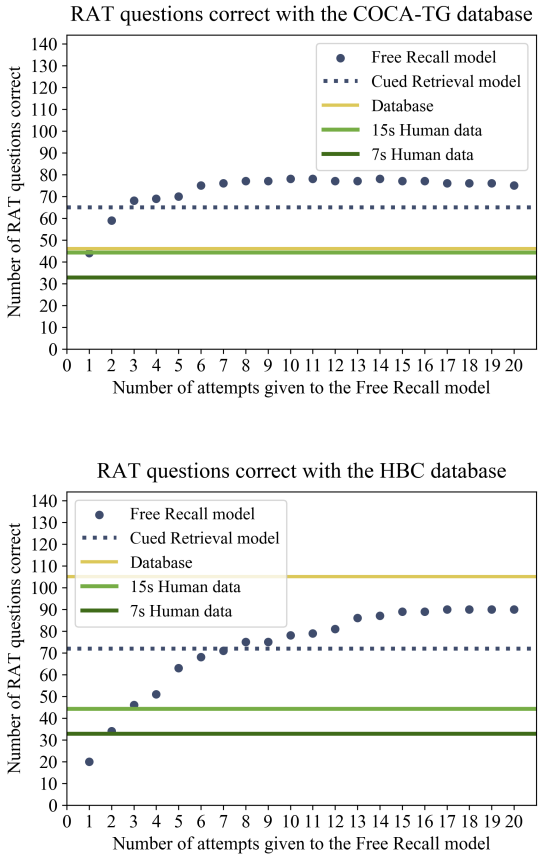


Figure 1: The number of RAT items each model with each database got correct out of the 144 possible items. Note the Free Recall model results are shown as 20 separate points.

configuration best matches human behavior. While our models generally perform better on the RAT than humans, we can separately characterize behavior by relative difficulty.

Timing Comparison In measuring timing, our goal is to see if our models are in the ballpark of human response times. A Soar model's timing can be roughly compared to human times. For Soar models, we consider a single decision cycle as corresponding to 50ms of human behavior. However, we consider retrieval as requiring roughly 300ms. Using those parameters, the Free Recall model using HBC and given 2 and 3 attempts took an average time of 2.02 and 2.20 seconds respectively to find a solution and the Cued Retrieval model took an average time of 2.27 seconds. The subjects in the study took on average 4.87 seconds when given 7 seconds to solve the problem and 7.26 seconds when given 15 seconds. Thus, the time taken for our models to solve a RAT item is similar in magnitude to how long it took the subjects. This rough similarity in timing suggests that our models are using approximately the same number of steps and retrievals as is found in human behavior.

Correctness Comparison In this section, we evaluate whether the RAT questions that are difficult for humans are also difficult for the models. For human difficulty, we focus on correctness and we use the percentage of people who got the correct answer as the metric of difficulty. For our models, we use whether the model produces the correct answer for a given item.

In order to compare these two metrics, we binned the 144 RAT items into 12 groups of 12 based on correctness in humans. The first group being the most difficult for humans (the lowest percentage of people got them correct), the last being the easiest (the highest percentage of people got them correct). We did this for both the 7 seconds and 15 seconds human results, as they had times closest to those predicted by our model. From the 12 questions in each bin, we calculated the mean percentage of people who got the questions correct. We then compare the average RAT items correct for humans to the number of items our models got correct for each 12 question bin. We did this comparison for the Cued Retrieval and the 1-20 guesses Free Recall models for each knowledge base. The correlations between the number of items correctly answered by humans and the number of items correctly answered by our models are shown in Figure 2.

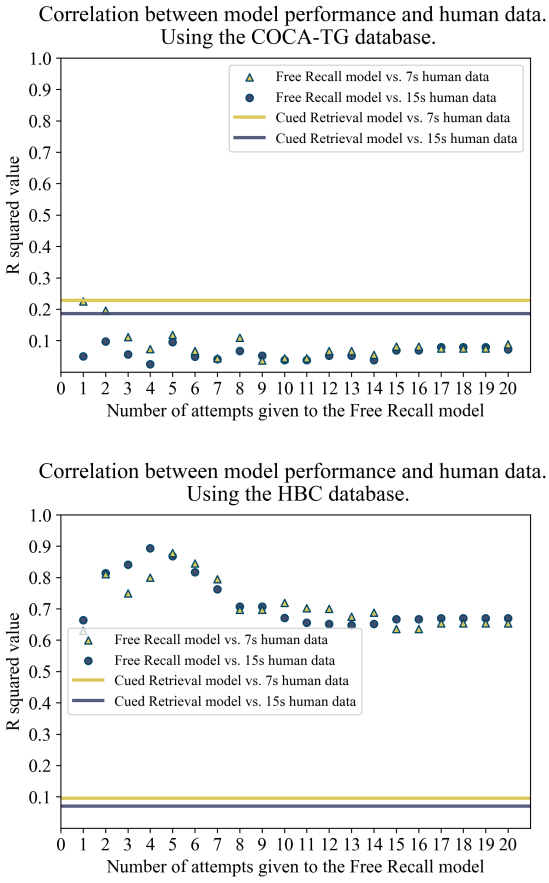


Figure 2: The correlations of model difficulty with human difficulty is displayed. The Free Recall model has a varying number of attempts displayed on the x axis.

The models using COCA-TG are shown in the top diagram and they have low correlations with human difficulty. Using COCA-TG, the highest correlated model is Free Recall with 1 attempt: 0.23. The models using HBC are shown in the bottom diagram, and all Free Recall model correlations are better using HBC. The highest correlated model is Free Recall with 4 attempts for the 15 second human data: 0.89. The Cued Retrieval model has low correlation for both databases. This suggests that HBC is a better model for the knowledge humans use to perform this task, and that the Free Recall model with 4 attempts is an excellent model of human difficulty.

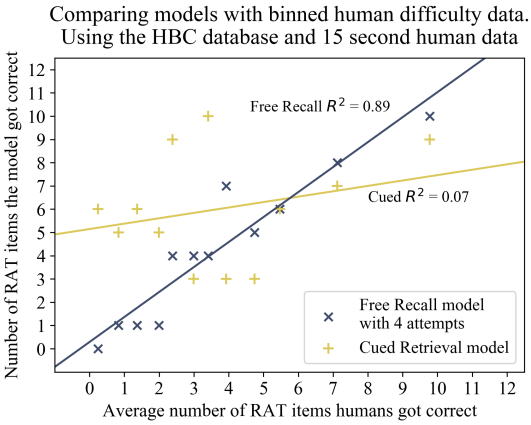


Figure 3: Agent performance is displayed with respect to human performance. The human data refers to the average number correct within a difficulty bin, for 12 bins. The best fit line is shown for both the Free Recall and the Cued Retrieval model data.

In Figure 3 we further investigate the most highly-correlating model configuration (Free Recall, 4 attempts) with a comparison to the Cued Retrieval model (both using HBC). This figure shows the number of RAT items each model got correct for each of the 12 groupings of items, ordered by difficulty, correlated with the expected number that humans got correct for each of those groups. The best fitting line has a slope of 1.074 and an y-intercept of 0.289 making it a close one-to-one relationship between the human's relative difficulty and the model's relative difficulty. To further verify our claim that the Free Recall model relates better to human data than the Cued Retrieval model, we ran a logistic regression test with the null hypothesis that the model's correct versus incorrect output for RAT items does not relate to the percentage of humans that got the RAT items correct. For the Free Recall model given 4 attempts, we reject the null hypothesis with a p-value of 2.86e-07. For the Cued Retrieval model we do not reject the null hypothesis, given a p-value of 0.184.

Fan and Association Strength Influence on Relative Difficulty Given a model and knowledge base which correlate with human relative difficulty, we next attempt to characterize the effects of association strength and fan on model difficulty. We selectively lesion the effects of fan and associ-

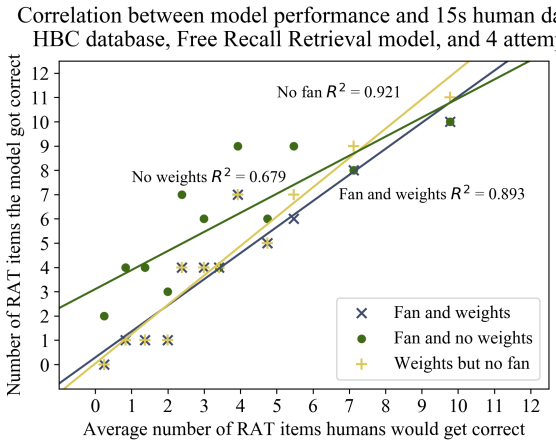


Figure 4: For our model configuration with the highest correlation to human relative difficulty, we also display models corresponding to the removal of association strength (weights) and fan.

ation strength on retrieval to show how the correlation with human difficulty changes as a result. Lesioning of fan leads to a model of spread where only association strength governs spread, and where there is no normalization with respect the number of outgoing links from the source. Lesioning of association strength leads to a model of spread where all association strengths from a given node are equal. These additional models are plotted with lines of best fit alongside our best matching model in Figure 4, where we again present model correctness compared to human correctness.

We expect that lacking association strength and only using fan should give better results in terms of absolute number correct because a single strong association can dominate during retrieval, whereas with equal strength for all associations, only those items that have associations with all given words will be retrieved. We confirm that the lesioned HBC with 4 attempts gets 72 RAT items correct versus the original HBC with 4 attempts which gets only 51 correct. Additionally, we expect that lesioning either fan or association weights should led to worse match to human difficulty. This is the case for removing weights, with the correlation dropping to 0.679, but removing fan improves the correlation to 0.921, suggesting a mismatch between the associations in our database and those in humans.

Conclusion

We created models that perform the remote associates test by employing two distinct methods. While a previous model for difficulty on this task did find association strength and fan to govern retrieval difficulty, our work provides a better account of how such influences impact difficulty by using a more realistic knowledge base and implementing our models as agents that complete the task, getting answers which can be directly compared to human answers. The Cued Retrieval model does a cued query to semantic memory to find the solution, if it exists. If a solution was not found, the model makes a plau-

sible guess. The Free Recall model iteratively uses spreading activation to retrieve a potential solution until it finds a solution or until it hits a threshold. The semantic memory knowledge bases only contained word associations. This is limited in comparison to a human's semantic memory. However, the use of spreading activation and the associations found in HBC's memory network give results surprisingly consistent in terms of relative difficulty for answering RAT problems with human performance. While we replicated the RAT-KB knowledge base associated with the previous work's model with our COCA-TG knowledge base, we found that despite it only consisting of the relevant type of associations for the 144 RAT problems, it performed worse than the HBC knowledge base in terms of modeling human difficulty. Our hypothesis is that a combination of inclusion of bidirectional links in COCA-TG leads the model astray by allowing it to find associations that are either missing or have very low association strength in humans.

From the results, we find that the Free Recall model with 4 attempts is an excellent match to relative problem difficulty in human behavior for when humans have 15 seconds for the task, although it is also highly correlated for a range of number of attempts. While the Cued Retrieval model can retrieve more answers (depending on the choice of attempt parameter for the Free Recall model), the Free Recall model using the HBC database has higher correlation with human results than the Cued Retrieval model with the same knowledge. This is seen with both the 7 and 15 seconds binned human data.

In attempting to characterize the role of fan and association strength in these results, we found that a better match to human difficulty is achieved when association strength but not fan influences spreading activation. One possible explanation is that there are artifacts in the HBC knowledge base in terms of missing items and their connectivity that do not reflect human semantic memory. We already know that they do not contain all the relevant knowledge for the RAT questions. Thus, we plan to expand the HBC database by adding other databases that include more of the relevant words and associations, while still being representative of human word associations, such as the University of South Florida Free Association Norms (Nelson, McEvoy, & Schreiber, 1998).

Another shortcoming of our databases is that they have no information about the recency and frequency of the words they include, and thus there is no contribution of base-level activation to our model (Anderson, Bothell, Lebiere, & Matessa, 1998). A reasonable extension would be to initialize our databases with usage information derived from other databases, such as COCA (Davies, 2008).

Acknowledgments

The work described here was supported in part by the Office of Naval Research under Grant Number N00014-18-1-2010. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the ONR or the U.S. Government.

References

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology*, *6*(4), 451–474.

Anderson, J. R. (2009). *How can the human mind occur in the physical universe?* Oxford University Press.

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language*, *38*(4), 341–380.

Anderson, J. R., & Pirolli, P. L. (1984). Spread of activation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *10*(4), 791.

Bowden, E. M., & Jung-Beeman, M. (2003). Normative data for 144 compound remote associate problems. *Behavior Research Methods, Instruments, & Computers*, *35*(4), 634–639.

Davies, M. (2008). *The corpus of contemporary american english*. BYE, Brigham Young University.

Derbinsky, N., & Laird, J. E. (2010). Extending soar with dissociated symbolic memories. In *Symposium on human memory for artificial agents, aisb* (pp. 31–37).

Gabler, K. (2013). *Human brain cloud.* Retrieved from `https://humanbraincloud.com/`

Griffiths, T. L., & Steyvers, M. (2002). A probabilistic approach to semantic representation. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 24).

Jones, S. J., Wandzel, A. R., & Laird, J. E. (2016). Efficient computation of spreading activation using lazy evaluation. In *Proceedings of the 14th international conference on cognitive modeling.*

Laird, J. E. (2012). *The Soar cognitive architecture*. MIT Press.

Mednick, S. (1962). The associative basis of the creative process. *Psychological Review*, *69*(3), 220-232.

Nelson, D. L., McEvoy, C. L., & Schreiber, T. A. (1998). *The university of south florida word association, rhyme, and word fragment norms.* `http://www.usf.edu/FreeAssociation/`.

Olteţeanu, A.-M. (2014). Two general classes in creative problem-solving? an account based on the cognitive processes involved in the problem structure-representation structure relationship. In *Proceedings of the international conference on computational creativity. publications of the institute of cognitive science* (Vol. 1).

Olteţeanu, A.-M., & Falomir, Z. (2015). Comrat-c - a computational compound remote associates test solver based on language data and its comparison to human performance. *Pattern Recognition Letters*, *67*, 81–90.

Olteţeanu, A.-M., & Schultheis, H. (2017). What determines creative association? revealing two factors which separately influence the creative process when solving the remote associates test. *The Journal of Creative Behavior*.

Modeling Decision Making in a Biased Matchmaker Task

Jaelle Scheuerman (jscheuer@tulane.edu)[1,2], **Dina Acklin**[1], **Noelle Brown**[1]
[1]Naval Research Laboratory, 1005 Balch Blvd, Stennis Space Center, MS 39556
[2]Department of Computer Science, Tulane University, 6823 St Charles Ave, New Orleans, LA 70118

Keywords: cognitive modeling; decision making; ACT-R; instance based learning theory

Introduction

Decision making involves choosing a course of action based on goals, knowledge, environmental cues and past experiences. This can be challenging because decision making often happens in environments where it is difficult to identify accurate and reliable information. This leads to relying on learned heuristics that can lead to systematic bias and sometimes costly errors (Kahneman & Tversky, 1972). Since people must often make decisions in complex environments, it is important to better understand the underlying cognitive mechanisms that lead to bias and develop strategies that may help mitigate errors. We present a cognitive model of a probabilistic learning task designed to assess bias reduction strategies, particularly attention to contradicting information and knowledge of erroneous feedback. This model utilizes instance based learning theory and a biased attentional weight parameter to model two groups completing the task.

Behavioral Task

In the behavioral task, subjects (N=203), were asked to play the role of a matchmaker and choose one of two bachelors for a given match, based on five attributes (hair color, hobby, entertainment preference, drinking habit and age).

Initially, subjects were biased to believe that an irrelevant factor (entertainment preference) was critical to choosing the correct bachelor. Instead, another factor (hair color) was critical to making accurate matches. Subjects responded to three types of trials: congruent, incongruent and irrelevant. Congruent and Incongruent trials paired the biased irrelevant factor with the critical factor. Irrelevant trials paired the critical factor with an entertainment preference that was not related to the participants' learned bias. Accepted responses were made for each trial type by matching based on the critical factor. If participants made matches based on their bias they were correct on Congruent trials and incorrect on Incongruent trials. Their bias was not useful on Irrelevant trials. Using these responses, it is possible to analyze how well subjects mitigated the bias and increased their match accuracy.

Subjects completed a Baseline phase of 30 trials, without feedback, to test whether or not a bias towards matching based on the irrelevant attribute had been established. Subjects then began a Learning phase, lasting 60 trials, where they received feedback about whether or not their match was accepted. To simulate making decisions in real world environments where information is not always accurate, the feedback was incorrect 25% of the time. Half of the subjects were warned that the feedback would sometimes be incorrect and the other half were not. Finally, the subjects completed a Testing phase of 30 trials to test whether or not they were able to overcome the initial bias to make accurate decisions. Results showed accuracy increasing throughout the task for both groups, with the Unwarned group outperforming the Warned group in the Testing phase as seen in Table 1.

	Baseline	Learning	Test
Unwarned	53%	63%	67%
Warned	52%	59%	61%

Table 1: Behavioral results for each group.

Methods

Using the cognitive architecture, ACT-R 7.0, we designed a cognitive agent to complete the matchmaking task described above. ACT-R consists of several modules that represent distinct cognitive systems, such as memory and perception (Anderson et al., 2004). ACT-R's learning mechanisms, combined with its declarative and procedural memory modules, make it well suited to modeling decision making tasks. Our agent also utilizes instance based learning theory and utility learning to complete the decision making task. We compare the cognitive agent's behavior to human subjects to test the underlying cognitive theories used in developing the model (Fum, Missier, & Stocco, 2007).

Instance based learning theory proposes that humans adapt their decision making strategies based on past experiences. Initially, they may use a heuristic approach, but as new situations are encountered, decision makers adjust their strategies to choose actions that previously resulted in favorable outcomes. When implementing an instance based learning model in ACT-R, situations are modeled as chunks stored in declarative memory, and include the situation, decision and any associated feedback. A production rule can be used to retrieve instances that are similar to the present situation, using ACT-R mechanisms such as partial matching and blending (Gonzalez, Lerch, & Lebiere, 2003).

Utility learning allows the agent to adapt its strategies to the environment (Anderson et al., 2004). ACT-R agents choose which actions to take based on production rules that reside in procedural memory. The agent selects rules by pattern matching against ACT-R's buffers that contain information available to the agent. These rules are retrieved based on an associated utility. When utility learning is enabled, the utilities adjust based on rewards given for retrieving specific production rules.

Cognitive Model

The cognitive model includes four production rules that govern how the agent completes the behavioral task. It simulates choosing the matching bachelor by using a heuristic that an incorrect feature (entertainment preference) is critical for making the decision. This is done by having a rule that assigns the match to Frank if the preferred entertainment is video games. A second rule assigns the match to James if sports is preferred. If neither video games or sports is preferred, a third rule randomly assigns the match to a bachelor. These rules emulate human behavior at the start of the task after being biased to believe that entertainment was critical to making a successful match.

In addition to the heuristic rules, a fourth rule assigns a match based on past experience, using the instance based learning approach. This rule looks for past decision making episodes stored in memory and chooses one with the highest activation value. The activation value is determined by how recently and frequently the previous instances have been retrieved. ACT-R's partial matching is enabled so that chunks matching the current situation to the greatest degree will have higher activation values. Additionally, spreading activation is used to represent the effect of context on the retrieval process. Here, the attentional weight of the entertainment attribute is a parameter that models how much attention is being given to the biased factor. The equations used to calculate the activation value with partial matching and spreading activation are well documented in ACT-R and IBLT literature (Anderson et al., 2004; Gonzalez et al., 2003).

The ACT-R agent completed 120 trials in the same order presented to human subjects. First, the agent completed 30 trials of a Baseline phase to establish that the heuristic rules successfully modeled a preference for making matches based on entertainment. After completing the Baseline phase, the agent proceeded to the Learning phase, which consisted of 60 trials. During this phase, the agent received feedback about about the match outcome that was incorrect 25% of the time. As soon as the model received feedback, the utilities of production rules updated so that the instance based learning rule had a higher utility than the heuristic rules. Upon completing the Learning phase, the agent advanced to the Testing phase, where no feedback was provided and the final decision making strategy was evaluated.

Model parameters

We completed a grid search to obtain values for the four parameters listed in Table 2 that best fit the behavioral data. Remaining ACT-R parameters were fixed at their default values. Noise (*ans*) was added to model the stochastic nature of the retrieval process. The mismatch penalty constant (*mp*) was adjusted to represent the degree of similarity that must exist between a chunk in the buffer and one retrieved from declarative memory. The retrieval threshold parameter (*rt*) represents how high the activation value must be for a chunk to be retrieved (Anderson et al., 2004). Finally, a parameter (*bw*) was adjusted to represent the amount of attentional weight applied to the entertainment attribute from spreading activation. For each combination of parameter values, the model was run 100 times to simulate 100 agents completing the behavioral task. The R^2 value was calculated between the model's average performance and the average performance of both groups of human subjects over all trials.

Results & Future Directions

The model fit the behavioral data relatively well, with the R^2 values summarized in Table 2. Our results showed the Warned group was best modeled when a higher attentional weight was applied to the irrelevant feature, whereas using a lower attentional weight resulted in a better fit for the Unwarned group. This seems to indicate that the Warned group had a more difficult time overcoming their bias towards the irrelevant feature. We hypothesize that since this group was warned that some feedback would be incorrect, they were less likely to trust feedback that contradicted their bias.

Model	ans	mp	rt	bw	R^2
Unwarned	0.75	2.25	2	1	0.9317
Warned	1.25	1.75	1	10	0.9470

Table 2: Best parameters and resulting R^2 values.

Further work must be done to understand how the modeled cognitive mechanisms can account for the increased accuracy observed in a small proportion of high performing subjects. It is expected that the attentional weight of the entertainment attribute should decrease over time to simulate mitigating bias and increased decision making accuracy. Future work will also explore how trust affects the sensitivity to feedback in overcoming bias.

Acknowledgments

The design of the agent was performed at the Naval Research Laboratory under funding number N0001417WX00111 from the Office of Naval Research to Noelle Brown of the Naval Research Laboratory.

References

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

Fum, D., Missier, D. M., & Stocco, A. (2007). The cognitive modeling of human behavior: Why a model is (sometimes) better than 10,000 words. *Cognitive Systems Research*, *8*, 135–142.

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

Kahneman, D., & Tversky, A. (1972). Subjective Probability: A Judgment of Representativeness. In *The Concept of Probability in Psychological Experiments* (pp. 25–48).

Towards a Physio-Cognitive Model of the Exploration Exploitation Trade-off

David M. Schwartz (dms061@bucknell.edu), Christopher L. Dancy (christopher.dancy@bucknell.edu)
Department of Computer Science, Bucknell University
701 Moore Avenue,
Lewisburg, PA 17837 USA

Keywords: exploration vs exploitation; utility, ACT-R/Φ, Project Malmo, reinforcement learning.

Introduction

Managing the exploration vs exploitation trade-off is an important part of our everyday lives. It occurs in minor decisions such as choosing what music to listen to as well as major decisions, such as picking a research direction to pursue. The dilemma is the same despite the context: does one exploit the environment, using current knowledge to acquire a satisfactory solution, or explore other options and potentially find a better answer. An accurate cognitive model must be able to handle this trade-off because of the importance it plays in our lives. We are developing physio-cognitive models to better understand how physiological and cognitive processes interact to mediate decisions to explore or exploit. To accomplish this, we utilize the ACT-R/Φ hybrid architecture (Dancy, 2013; Dancy et al., 2015) and the Project Malmo AI platform (Johnson et al., 2016).

Modelling the Trade-off

ACT-R/Φ

ACT-R/Φ creates a representation of physio-cognitive mediation of behavior by combining the ACT-R theory of cognition, HumMod's physiological model (Hester et al., 2011) and theory from affective neuroscience. This hybrid architecture allows us to model how the management of the exploration versus exploitation trade-off effects the body and mind. Furthermore, the architecture provides a more concrete and tractable method to interact with the model by utilizing concentrations of hormones in the system to influence behavior. Changes in arousal, utility, and decision making can be seen through modifications of hormone concentration and regulation, providing an in depth look at how and why the trade-off is managed.

Model Assumptions

The model makes several assumptions to interact with the task environment. First, the model assumes it is in a diminishing return environment. Second, that cues are present in the environment, which provide information about the task. Lastly, that the agent is striving towards some goal.

Managing the Trade-off

The high-level model manages the exploration exploitation trade-off according to the abstract rules in Figure 1.

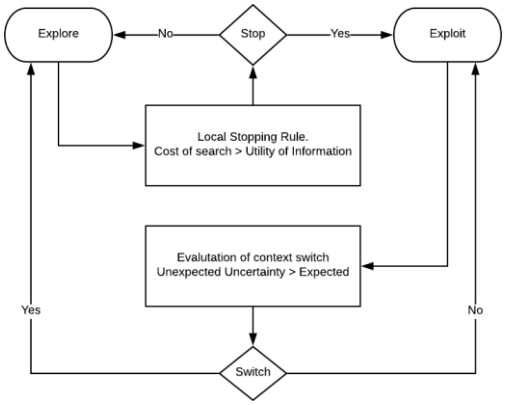


Figure 1. The high-level decisions the model makes to transition between exploring and exploiting.

The model decides to stop exploring via a local stopping rule that integrates the Satisficing Model by Wai-Tat Fu (2006). The model assesses the utility of information, that is, how useful the new information gained from exploring is, against the cost of further exploration. Since the model assumes it's in a diminishing return environment, the cost of searching will be lower at the start and increase as time goes on. Thus, exploration will tend to occur early. Once the cost of search outweighs the benefit of information gained the model will decide how to exploit its knowledge and will continue to exploit until it deems the current method is no longer adequate.

The decision to return to exploration is controlled by assessments of expected and unexpected uncertainty. When unexpected uncertainty is higher than expected, the current method of solving the problem is no longer reliable, thus exploration should start. Yu and Dayan (2005) related these concepts to the neuromodulators acetylcholine (ACh) and norepinephrine (NE). In their formulation, ACh represents expected uncertainty and NE represents unexpected uncertainty. They also developed an equation that relates the concentration of the modulators to the choice to explore (Equation 1).

$$NE > \frac{ACh}{0.5 + ACh} \quad (1)$$

The equation represents low level self-assessment and triggers the transition between exploiting and exploring.

When it is satisfied, subsequent drops in utility and arousal are observed, denoting a loss of faith in the current strategy and need to discover a new one.

The decision making transition is reflected in the model by dynamically modifying the utility noise parameter, also referred to as temperature, in ACT-R. ACT-R selects which production to fire by its utility value. However, those values contain noise. The parameter controls the standard deviation of noise within the system. As noise increases, the probability that the production with the highest reinforcement will be selected decreases. Therefore, the likelihood of the model selecting another, less reinforced, production that satisfies that same scenario increases, leading to exploratory behavior.

As the model receives rewards, the temperature decreases, making production selection more deterministic. This results in the model switching back to the exploiting state. While the model runs, temperature is adjusted, becoming lower when search costs outweigh the value of current information, and larger when self-assessments reveal poor performance.

Testing the Model

We are using a symbolic maze, similar to the one used by Fu and Anderson (2006), to test the model. The structure of the maze is depicted in figure 2.

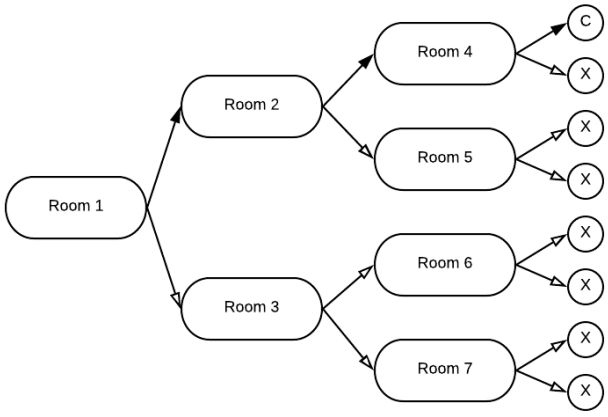


Figure 2. Structure of the symbolic maze. X represents a dead end whereas C depicts the exit. Picture based on an image from Fu and Anderson (2006).

The player is placed in a room and presented with stimuli and a set of options. They move into a different room depending on which option they select. Upon reaching a dead end the player is reset to the point where they diverged from the correct path. Furthermore, the configuration of the room is changed; different stimuli and options are shown upon their return. Thus, the player is only informed of correct stimuli option associations upon completing the maze or reaching a dead end.

The maze is implemented in Microsoft's Project Malmo environment. Project Malmo is a modification to the game Minecraft that allows artificial agents to be tested. The world is represented as a series of semantically defined blocks. This works well with ACT-R based models as the representations in the perceptual modules are semantic attentional chunks. Thus, transforming the *blocks* to *chunks* is straightforward. For our experiment, stimuli is represented by special blocks in a wall. Decisions are made by standing on one of two sections of ore in the floor. After a decision is made, the player or agent is teleported to another room and the experiment continues as previously described.

Another benefit of using Project Malmo is its expandability. The tool can be used to construct varied environments with differing complexities from the same primitives. Using this platform allows us to modify the task to study different aspects of physio-cognitive mediation of human behavior in future work.

Conclusion

Managing the trade-off between exploration and exploitation is a critical part of our everyday lives. Our goal is to develop a model that manages the problem like a human does. We manage the transition from exploration to exploitation by assessing the cost of searching with the utility of information gained. The model handles the inverse transition by low level self-assessments of uncertainty, both expected and unexpected, within the problem. In addition, by using Project Malmo, we have created a useful, modifiable, task environment for future cognitive models. By improving our model to tackle more complex domains within Project Malmo we will be one step closer to developing human-like autonomous artificial agents.

References

Dancy, C.L (2013) ACT-RΦ; A cognitive architecture with physiology and affect. *Biologically Inspired Cognitive Architectures*, 6(1), 40-45.

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. (2007). A Rational-Ecological Approach to the Exploration/Exploitation Trade-offs. In W.D. Gray (ed.), *Integrated Models of Cognitive Systems* (Vol. 1, pp 165-179). New York, NY: OUP.

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

Hester, R. L., Brown, A. J., Husband, L., Iliescu, R., Pruett, D., Summers, R., & Coleman, T. G. (2011). HumMod: A modeling environment for the simulation of integrative human physiology. *Frontiers in physiology*, *2*(12).

Johnson, M., Hofmann, K., Hutton, T., & Bignell, D. (2016). The Malmo platform for artificial intelligence experimentation. *In proceedings of Twenty-Fifth International joint conference on artificial intelligence (IJCAI)*, New York, NY, 4246-4247.

Yu, A. J, & Dayan, P. (2005). Uncertainty, Neuromodulation, and Attention. *Neuron, 46*(4), 681-692.

Deploying a Model-based Adaptive Fact-Learning System in University Courses

Florian Sense (f.sense@rug.nl), Maarten van der Velde (m.a.van.der.velde@rug.nl), & Hedderik van Rijn (d.h.van.rijn@rug.nl)
Department of Experimental Psychology & Behavioral and Cognitive Neuroscience
University of Groningen, Groningen, The Netherlands

Abstract

Effective adaptive learning systems are based on computational models of learning and forgetting in human memory. These are often developed and validated in laboratory settings. Recently, the ACT-R-based model developed in our lab (Sense, Behrens, Meijer, & van Rijn, 2016) was made available to students enrolled in two Cognitive Psychology courses at the University of Groningen as an optional tool to study and rehearse material. Here, we provide an overview of data recorded throughout the course (including exam performance), as well as a preview of explorations and analyses made possible by the data. Specifically, such data allow exploration of two questions: (1) How do students use the system?, and (2) How does the system perform "in the wild"? Findings pertaining to (1) will be interesting to compare with survey responses, while findings pertaining to (2) can shed light on ways to improve the system in realistic educational settings.

Keywords: Model-based adaptive learning; real-world application; fact-learning; memory models; ACT-R.

Introduction

Individuals differ in their ability to acquire new knowledge. A theoretical understanding of human memory facilitates quantifying and understanding the individual differences in the temporal dynamics of learning and forgetting. ACT-R's declarative memory, for example, is based on formalizing regularities in behavioral data of learning and forgetting. Once formalized, such a system can be used to make predictions about which learner is going to forget what and when, which constitutes the basis for an adaptive learning system (Pavlik & Anderson, 2008).

The development and validation of adaptive fact-learning systems is an active area of research. The majority of the scientific literature on this topic is based on using such systems to test specific assumptions about human memory (e.g., Mettler, Massey, & Kellman, 2016), how these systems compare to various control conditions (e.g., Lindsey, Mozer, Cepeda, & Pashler, 2009), or the mechanics of the systems themselves (e.g., Mettler, Massey, & Kellman, 2011). The model developed in our lab is no exception (e.g., Sense et al., 2016; van Rijn, van Maanen, & van Woudenberg, 2009). Studying the memory mechanisms underlying human learning processes is obviously important, since it provides the foundation upon which such systems are built.

Most studies present participants with relatively arbitrary learning materials in a controlled lab-based setup. Herein we present a different testbed for an adaptive system: We let students use the system in unconstrained, realistic conditions. The system was made available to students enrolled in Cognitive Psychology courses at two departments (Psychology and Artificial Intelligence) at the University of Groningen in the academic year 2017/2018. The same textbook is used in both courses and material is grouped by chapter and made available in the week that each chapter is discussed in the course[1]. Using the adaptive system is optional and we ask students to give informed consent for their data to be used. Specifically, consent is asked to link the data recorded while using the adaptive system to their performance on the exam administered at the end of the course. Importantly, the exams of both courses will include a subset of questions that are taken from the material that was available for students to study with the adaptive system throughout the course. The data thus obtained provide a unique possibility to test an adaptive fact-learning system "in the wild". Details about the recorded data and proposed analyses and explorations are outlined below.

The Recorded Data

The adaptive system is available to students in the Blackboard environment through any web browser and as an app for both iOS and Android, which means that students can study when- and wherever they choose. Data will be available from two cohorts:

Cognitive Psychology is a third-year elective course in the **Psychology** undergraduate program. A total of 285 students took the exam (first attempt) and of those, 147 gave informed consent for their data to be used (51.6%). The grades are linked with the data generated while using the adaptive fact-learning system, which was used at least once by 137 students (52.5% of all recorded users). Together, they contribute 317,779 individual trials (65.2% of all recorded trials).

In the **Artificial Intelligence** undergraduate program, Cognitive Psychology is a mandatory first-year course. Currently, 162 students are enrolled in the program and roughly 90% of them are expected to take the exam, which is scheduled for early April. How many will consent for their data to be used is unknown at this point.

[1] In the course taught at AI, students also generated multiple-choice questions themselves as part of homework assignments. These questions were gathered and made available to all students for rehearsal/study with the adaptive system.

Proposed Analyses and Explorations

On the poster, we will present data from two cohorts. The presentation will focus on two overarching questions, which we will discuss below[2].

How do students use the system?

In both cohorts, study material for the adaptive system is made available progressively throughout the course. Ideally, students would start rehearsing this material as soon as possible to space out their repetitions as much as possible but survey data suggests that students cram their study time shortly before the exam (Dunlosky, Rawson, Marsh, Nathan, & Willingham, 2013). The collected data allows us to see when students chose to engage with the adaptive system and whether their behavior matches previously reported survey data. Additionally, we can determine whether students that start learning early (or: have wider spacing) obtain higher grades, which is an outcome measure more directly relevant to most students than the potential for increased long-term retention.

Students can pick the desired duration at the beginning of each study session. It will be interesting to see the distribution of study session durations and how those sessions are distributed within a day.

How does the system perform "in the wild"?

There are a number of ways to evaluate the system's performance. Importantly, all of the factors discussed here will necessarily be confounded with students' general motivation[2]. Nevertheless, it will be interesting to see which aspect of using the adaptive system are predictive of exam performance (if any). Importantly, ways in which the model fails and can be improved can be identified regardless of students' motivation.

Studying with the system is entirely optional. Testing whether students that choose to use the system score higher grades on the exam is a logical first step. More interesting, however, will be an analysis of the items on the exams that were available as study material prior to the exam (see above for details): Even if a student used the system, they might not have seen all the items. And even if they did, there will be variation in how often each item has been repeated and in the estimated model parameters, for example. Using performance measures of this kind obtained while studying with the system and testing whether they predict item-level exam performance will be very interesting.

An additional line of exploration will be devoted to comparing the recorded behavior with the predictions made by the system. Using the response history for a given item, the model generates trial-by-trial predictions for retrieval accuracy and latency (see Sense et al., 2016 for details). These can be compared with the recorded responses, which are expected to differ drastically from responses collected during controlled lab experiments conducted previously. Students can use the app to study anytime and anywhere, which is expected to result in much "messier" data. Ultimately, however, adaptive systems like this should function in such realistic settings so this data will be an excellent benchmark and reveal ways in which the system needs to be improved further to make it more useful to students.

[2] A general disclaimer for all ideas outlined in this section is that performance measures and usage statistics will be confounded with motivation: Using the system more might be linked to higher grades, for example, either because motivated students use the system more or because using the system results in higher grades. The options for correcting for motivation as a confound are – unfortunately – limited, and causal claims are categorically impossible to make.

Acknowledgments

We would like to thank Tom Doesburg for programming help, and Fokie Cnossen and Maximillian Velich for helping collect the data from the Artificial Intelligence sample. This work was supported by an E-Learning Grant provided by the University of Groningen, awarded to Van Rijn.

References

Dunlosky, J., Rawson, K. A., Marsh, E. J., Nathan, M. J., & Willingham, D. T. (2013). Improving Students' Learning With Effective Learning Techniques: Promising Directions From Cognitive and Educational Psychology. *Psychological Science in the Public Interest*, *14*(1), 4–58.

Lindsey, R., Mozer, M., Cepeda, N. J., & Pashler, H. (2009). Optimizing Memory Retention with Cognitive Models. In *Proceedings of the Ninth International Conference on Cognitive Modeling (ICCM 2009)* (pp. 74–79). Manchester, UK: ICCM.

Mettler, E., Massey, C. M., & Kellman, P. J. (2011). Improving Adaptive Learning Technology through the Use of Response Times. In L. Carlson, C. Hoelscher, & T. F. Shipley (Eds.), *Proceedings of the 33nd Annual Conference of the Cognitive Science Society* (pp. 2532–2537). Boston, MA: Cognitive Science Society.

Mettler, E., Massey, C. M., & Kellman, P. J. (2016). A comparison of adaptive and fixed schedules of practice. *Journal of Experimental Psychology: General*, *145*(7), 897–917.

Pavlik, P. I., & Anderson, J. R. (2008). Using a Model to Compute the Optimal Schedule of Practice. *Journal of Experimental Psychology. Applied*, *14*(2), 101–117.

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.

van Rijn, H., van Maanen, L., & van Woudenberg, M. (2009). Passing the Test: Improving Learning Gains by Balancing Spacing and Testing Effects. *Proceedings of the 9th International Conference on Cognitive Modeling*, 110–115.

Toward a Theory of Timing Effects in Self-Organized Sentence Processing

Garrett Smith (garrett.smith@uconn.edu)
Department of Psychological Sciences, 406 Babbidge Road, Unit 1020
Storrs, CT 06269 USA

Whitney Tabor (whitney.tabor@uconn.edu)
Department of Psychological Sciences, 406 Babbidge Road, Unit 1020
Storrs, CT 06269 USA

Abstract

Many theories of sentence processing are based on the idea that a discrete, symbolic grammar defines all of the structures relevant for parsing, effectively supervising the parser as it selects from those structures the one that best fits the input. However, *local coherence effects*, where people's parsing behavior suggests they are entertaining locally viable but globally impossible structures, suggest that this may not always be the case. We introduce a self-organized sentence processing (SOSP) model of local coherence effects and use it to demonstrate how predictions about timing effects (a major source of psycholinguistic data and a shortcoming of many previous dynamical parsers) can be derived directly from a harmony (well-formedness) function covering both grammatical and ungrammatical structures. This framework allows us to simulate the processing of any set of lexical features and attachment links, making it widely applicable to psycholinguistic phenomena.

Keywords: sentence processing, local coherence effects, dynamical systems models, self-organization

Introduction

The current, most fully-developed models of online sentence processing adopt an assumption which may be called *grammar supervision*. With grammar supervision, a symbolic grammar specifies the universe of structures possible for language comprehension and production, and the parser only considers those grammatical structures. An example is surprisal theory (Hale, 2001; Levy, 2008), in which the parser distributes probability over all grammatical structures compatible with the current input at each word. The processing time for each word is proportional to how much change in the probability distribution is needed after incorporating a new word (the Kullback-Leibler divergence between prior and posterior distributions estimated from a large corpus). This kind of theory has been massively successful in modeling reading times in both experimentally designed stimuli and natural corpora (Levy, 2008; N. J. Smith & Levy, 2013).

However, empirical studies over the past several decades have identified a number of phenomena that challenge the grammar-supervision hypothesis. We focus on local coherence effects (Ex. (1); Bicknell, Levy, & Demberg, 2009; Konieczny, Müller, Hachmann, Schwarzkopf, & Wolfer, 2009; Kukona, Cho, Magnuson, & Tabor, 2014; Levy, Bicknell, Slattery, & Rayner, 2009; Paape & Vasishth, 2015; Tabor, Galantucci, & Richardson, 2004). Early-arriving words make it so that, if the grammar were supervising, only one parse would be possible, but when later words are perceived, people show evidence of entertaining a second, conflicting parse motivated by the later-arriving words. For example, the reduced forms in of Ex. (1) (i.e., without *who was*) showed slowed reading at *tossed/thrown* relative to the unreduced form, but this effect was significantly larger for (1-a) than for (1-b) (Tabor et al., 2004).

(1) a. The coach smiled at the player (who was) tossed the Frisbee by the opposing team.
b. The coach smiled at the player (who was) thrown the Frisbee by the opposing team.

We can make sense of this result if we assume that the words *the player tossed...* (but not *thrown*) cause the parser to construct an active clause with *the player* as its subject, even though English grammar mandates that, in this context, *tossed* be a passive verb heading a reduced relative clause modifying *the player*. This process is inconsistent with grammar-supervision theories, but it is naturally predicted if parsing is governed by principles of self-organization.[1]

Self-organized sentence processing (SOSP; Kempen & Vosse, 1989; Stevenson, 1994; Tabor & Hutchins, 2004; van der Velde & de Kamps, 2006; Vosse & Kempen, 2000, 2009; Cho et al., 2017; G. Smith, Franck, & Tabor, 2018; Gerth & beim Graben, 2009)) is an approach to modeling sentence processing which does not assume grammar supervision. Instead, in analogy to many physical chemical and biological processes (see, e.g., Haken, 1983), parses self-organize (without any controller or external supervision) via continuous, local, bottom-up interaction among small pieces of syntactic tree structure (treelets) activated by the words that have been perceived or are being produced. In SOSP, feedback interactions among the treelets generally drive the formation of structure consistent with the grammar, but when two or more incompatible structures receive bottom-up support, the system can stabilize in an ungrammatical state of conflict, causing processing difficulty. Such models have produced plausible accounts of center embedding vs. right branching, garden path effects, lexical ambiguity processing (Vosse & Kempen, 2000), length effects (Tabor & Hutchins, 2004), and agreement attraction (G. Smith et al., 2018), among others.

[1] Levy et al. (2009) argue that surprisal can account for Tabor et al. (2004) with a *noisy channel* assumption—words may be misperceived (e.g., *at* was actually *and* in Ex. (1-a)). Cho, Goldrick, and Smolensky (2017) present a similar approach in a dynamical model. However, not all local coherence effects are plausibly amenable to this explanation (Kukona et al., 2014; Paape & Vasishth, 2015).

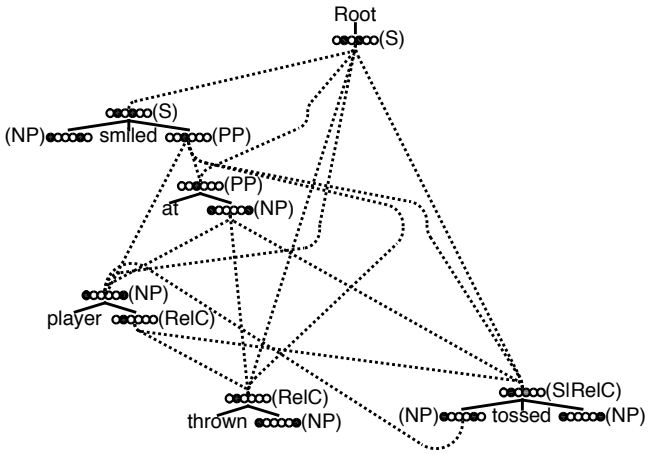


Figure 1: A snap-shot of SOSP-TH parsing a fragment of Ex. (1) showing a subset of competitive treelet interactions. Circles represent features (in order: Nominal, Verbal, Prepositional, Matrix-Clause, Agent, Patient) on attachment sites (labeled in parentheses); phonological forms are unmarked; and the dotted lines are attachment links. Note that even ill-formed structures are included, e.g., *tossed* attaching to Root as the matrix verb instead of the relative clause (RelC) head.

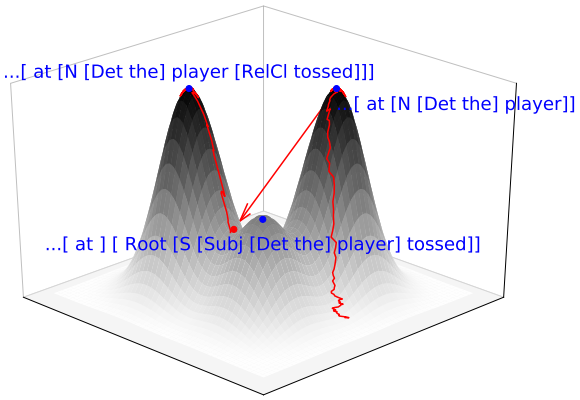


Figure 2: A partial harmony surface illustrating a sample processing path. The vertical axis is harmony, and the other dimensions code feature/link configurations. After reading *the coach smiled at the player*, the noisy dynamics push the system toward a partial parse with *the player* attached as the nominal dependent of *at* at the peak labeled [*at* [N [Det *the*] *player*]]. After stabilizing there, *tossed* is read, jumping the system (red arrow) to a point intermediate between the grammatical [*at* [N [Det *the*] *player* [RelCl *tossed*]]] and the locally coherent, low-harmony [*at*] [Root [S [Subj [Det *the*] *player*] *tossed*]] (with *at* not attached to the subsequent words). From there, the system settles again, in this case selecting the grammatical peak.

Oddly, there are relatively few SOSP results on timing data, even though timing data are the most common kind of psycholinguistic data, and even though self-organization is generally understood via dynamical systems theory, the mathematics of variables interacting in time. Our main contribution here is a novel SOSP framework that addresses this shortcoming by making the relationship between well-formedness and processing times transparent. Influenced by Cho et al. (2017), Smolensky (1986), and Haken (1983), we define a harmony function (also known as a potential or energy function) that specifies the global well-formedness of system states (configurations of features on attachment sites and attachment links, Fig. 1). We employ a systematic method of deriving the harmony function from lexical features in parsed sentences, creating a hilly landscape with peaks corresponding to both fully grammatical structures and conflict states (Fig. 2). The sentence processing dynamics noisily push the system uphill on this landscape to find local harmony maxima. This leads to a theory of timing effects in which, all other things being equal, a higher-harmony parse is built faster than a lower-harmony one. This is because higher peaks have steeper gradients, causing the system to move faster toward the peak. In ambiguous sentences, the system stochastically selects among different peaks, and its path will be more curved if competing peaks are more equally well-formed. Therefore, average processing times over many trials depend on which peaks are selected and how curved the trajectories are.

Below, we present our SOSP framework (called SOSP-TH ("treelet harmony") to distinguish it from other SOSP models), show how it makes timing predictions, report an implemented SOSP-TH model of local coherence, and finally discuss SOSP-TH in relation to other psycholinguistic theories.

The SOSP-TH framework

In SOSP-TH, linguistic structures are built out of lexically anchored syntactic treelets that connect with each other via graded attachment links (Fig. 1). We assume for simplicity a dependency grammar formalism (e.g., McDonald et al., 2013), so the only attachment sites are ones linking a word as the dependent of another word (*head* attachment sites) and ones linking other words as dependents (*dependent* attachment sites). The head and dependent attachment sites are feature vectors encoding syntactic and semantic properties of a word and its expected dependents, respectively. Some features can change (e.g., the determiner *the* gets its number marking from its licensor), while others are fixed in the lexicon. The only constraints on link formation are that 1) no links can form within a single treelet (e.g., a determiner dependent site on a noun cannot link to the head of that same noun) and 2) links can only form between head attachment sites and dependent attachment sites, i.e., no head-head or dependent-dependent links.[2]. All other links, grammatical and ungrammatical, are allowed to form. Finally, a special

[2]Links may to fail to form, making fragmentary, low-harmony parses.

root node is available to anchor the whole sentence.

Features and links that are fully "on" and "off" are coded as 1 and 0, respectively. In order to allow multiple tokens of the same treelet in one sentence (e.g., *the* in *the dog saw the cat*), all of a treelet's dimensions are repeated for every position in a sentence. Thus, there is a set of dimensions corresponding to *the* as the first word of a sentence, a different set of dimensions for *the* as the second word, etc. Links (additional dimensions of the system) are between sentence-position-specific instances of treelets.[3]

Not all attachment links make equally well-formed structures, though. Structures in which all linked feature vectors are perfectly matched receive the maximum harmony of 1. Any feature mismatch lowers the harmony for that structure. In this way, SOSP implements a graded notion of well-formedness. We quantify the local harmony h_i of a (partial) linguistic structure i, i.e., degree of well-formedness for i's configuration of features and links, using Eq. 1:

$$h_i = \prod_{l \in links} \left(1 - \frac{dist(\mathbf{f}_{l,head}, \mathbf{f}_{l,dependent})}{nfeat}\right) \quad (1)$$

The local harmony h_i of a structure is the product of one minus the normalized Hamming distances $dist(\cdot)$ between the head feature vectors $\mathbf{f}_{l,head}$ and dependent feature vectors $\mathbf{f}_{l,dependent}$ for each link l. $nfeat$ is the number of elements in the feature vectors. This definition of local harmony is valid for any combination of features and links, even those that strongly violate rules of a symbolic grammar, e.g., the fragmentary, locally coherent structure [at] [Root [S [Subj [Det the] player]] tossed]. In the simulations below, we will see that including these lower-harmony structures in the mental representation of possible structures plays a key role in explaining observed timing effects.

Eq. 1 allows us to calculate the harmony of any linguistic configuration, but on their own, the h_is do not tell us how to choose a structure given the input. To that end, we define a global harmony function and derive the dynamics from it.

Defining the harmony landscape and dynamics

We can define where the peaks in our harmony function are by using a sum of radial basis functions (RBFs) ϕ_i (Han, Sayeh, & Zhang, 1989; Muezzinoglu & Zurada, 2006):

$$\phi_i(\mathbf{x}) = \exp\left(-\frac{(\mathbf{x}-\mathbf{c}_i)^\intercal(\mathbf{x}-\mathbf{c}_i)}{\gamma}\right)$$

Here, $\mathbf{x}$ (a column vector) is the d-dimensional state of the system encoding values of all features and links in $\mathbb{R}^d$, each $\mathbf{c}_i$ is the location of the ith (partial) parse (encoding desired feature values and link strengths), $^\intercal$ denotes the vector transpose[4], and γ (a free parameter) sets the width of the RBFs. We then define the harmony function $H(\mathbf{x})$ as the sum of n RBFs, where n is the number of partial and full parses (harmony peaks) we wish to encode:

$$H(\mathbf{x}) = \sum_i^n h_i \phi_i(\mathbf{x}) \quad (2)$$

where the h_i give the local harmony of a (partial) parse, computed using Eq. 1. This equation creates a hilly harmony landscape analogous to Fig. 2, assigning harmony values both to the $\mathbf{c}_i$ and to all states intermediate between them.

In SOSP-TH, treelets are interacting subsystems that attempt to assemble themselves through local interactions that locally maximize harmony. Since the gradient of a scalar-valued function like $H(\mathbf{x})$ points in the direction of steepest ascent, we make the system change in time so that it follows this gradient uphill in a noisy way:

$$\frac{d\mathbf{x}}{dt} = \nabla_{\mathbf{x}} H(\mathbf{x}) = -\tfrac{2}{\gamma}\sum_i^n h_i(\mathbf{x}-\mathbf{c}_i)\phi_i(\mathbf{x}) + \sqrt{2D}\, dW \quad (3)$$

(D scales the magnitude of the Gaussian noise process dW). For $D = 0$, gradient dynamical systems like this simply settle from an initial condition to an attractor (points to which the system will return after a small perturbation; Strogatz, 1994). For $D > 0$, the noise helps determine which attractor the system converges on.

Any parsed corpus can be represented as a set of vectors (the $\mathbf{c}_i$) of lexical features at particular sentence positions and links between attachment sites, making SOSP-TH a general theory of sentence processing. Note that once the $\mathbf{c}_i$ are specified, the harmony landscape does not change, unlike in the Gradient Symbolic Computation framework (Cho & Smolensky, 2016; Cho et al., 2017; Cho, Goldrick, Lewis, & Smolensky, 2018), in which the harmony function changes with the input. Since the parsing dynamics are derived directly from the harmony function, the SOSP-TH parser is derived directly from a parsed corpus of sentences. We now show how we can derive processing time predictions from these equations.

Predicting processing times

To derive predictions about processing times, we first consider the simplest possible case, a one-dimensional system with a single harmony peak at $x = 0$. The harmony function is $H(x) = h\,\phi(x) = h\,\exp\left(-\frac{x^2}{\gamma}\right)$ and the dynamics are given by $\dot{x} = -\frac{2h}{\gamma}\, x\, \phi(x)$. From this equation, we can already see that the higher the harmony of the attractor, the faster system moves toward it: Well-formed structures are faster to build than ill-formed structures.[5]

In general, though, an SOSP-TH parser will have many dimensions coding multiple features and link strengths, and

[3] This parallels the TRACE model of word perception (McClelland & Elman, 1986), where every position of every word is a node in the model. We agree with the critique that this is neurally implausible and may miss important generalizations. However, TRACE has been very successful at capturing phonological effects in word processing, so we feel this is a reasonable place to start.

[4] $(\mathbf{x}-\mathbf{c}_i)^\intercal(\mathbf{x}-\mathbf{c}_i)$ is the squared Euclidean distance between $\mathbf{x}$ and $\mathbf{c}_i$.

[5] There are other ways to show how settling times in a single trial depend on the harmony of the parse that forms. One is to consider the time dt it takes to travel an infinitesimal distance dx, $dt = dx/\dot{x}$, since time equals distance divided by velocity. Integrating both sides shows the settling time $t \propto (2h)^{-1}$. A third option, linear stability analysis (Strogatz, 1994) provides a similar result.

there will be many attractors corresponding to different structural alternatives. To see that higher harmony still means faster processing, we can approximate Eq. 3 near an attractor i by neglecting all terms $j \neq i$ in the sum in Eq. 3, as the effect of all other attractors drops off exponentially: $\dot{\mathbf{x}} \approx \frac{-2h_i}{\gamma}(\mathbf{x} - \mathbf{c}_i)\phi_i(\mathbf{x})$. It is clear that the same relation between settling time and harmony holds. However, the effects of other attractors are, in general, not completely negligible. Fig. 3 shows how the presence of a relatively high-harmony competitor can bow trajectories away from an attractor by warping the harmony landscape, even though the system is not in the basin of attraction of the competitor.

Thus, the overall theory of timing effects in SOSP-TH is this: Within a basin of attraction of a structure, the settling time scales approximately inversely proportional to the harmony of that parse, modulo the noise and the bowing. Over repeated trials, noise will bump the system toward attractors of different harmony heights, so the average settling time at a word is the average of the settling times to each selected attractor weighted by how often the attractor is selected. We now illustrate this in a simple model of local coherence.

An SOSP-TH model of local coherence effects

A full model of the incremental processing of the sentences in (1) would involve incrementally turning on features of words in their sentence positions, letting the system settle to an attractor associated with a partial parse, and repeating until the sentence ends (see Fig. 1). We can model the main local coherence finding from Tabor et al. (2004) in a focused way by assuming that the parser has already read up to *The coach smiled at the player tossed/thrown...* and that it must now choose how to attach *player* and *tossed/thrown*. We need only two dimensions, one for the grammatical *player-tossed* link and one for the locally coherent *tossed*-Root link. There is thus an attractor at [1, 0] (local harmony $h_0 = 1.0$) and one at [0, 1], which will have different sub-maximal harmonies (h_1) depending on whether *tossed* or *thrown* has been read (see Fig. 3). *Player* is a good feature match to be the subject of *tossed*, and *tossed* can function as a main verb attaching to the root node, so the attractor at [0, 1] is penalized only for leaving *the coach smiled at* unattached to the rest of the structure. For *thrown*, though, [0, 1] is additionally penalized because *thrown* cannot function as a main verb, so its features do not match Root's main-verb dependent features. We start the system at [0, 0], not biased toward either attractor.

SOSP-TH predicts that the noise should bump the system toward the grammatical parse in most cases because its high harmony dominates the harmony landscape. When the noise does push the state toward the locally coherent attractor, it will approach it more slowly in the *thrown* condition than in the *tossed* condition because of *thrown*'s especially low harmony. But because this happens so rarely, the average time will be dominated by fast approaches to the grammatical attractor. The locally coherent parse for *tossed* will be selected more often due to its higher harmony, so it will increase the average settling time more than *thrown*. There is also more trajectory bowing for *tossed*, which also slows processing (Fig 3). Thus, a relatively high-harmony competitor for the grammatical parse will, on average, cause a competition-based slowdown.

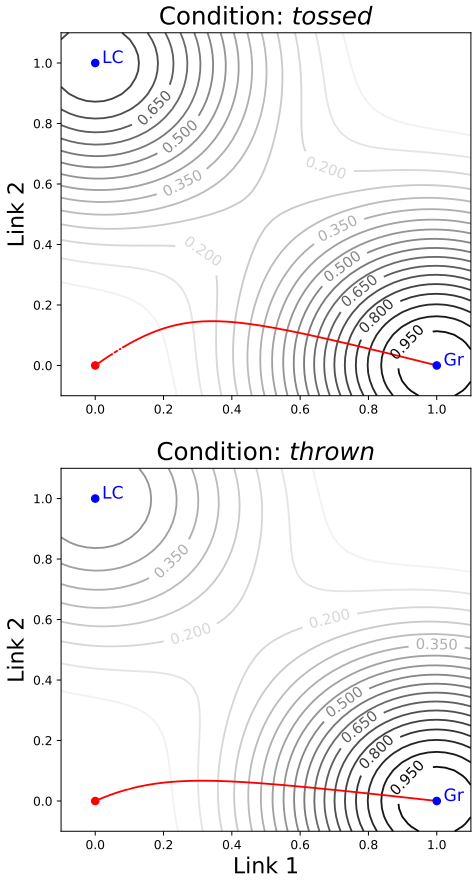


Figure 3: Contour plots of the harmony landscapes used in the local coherence simulations. Contour labels give the harmony at that level. Red lines show noiseless trajectories starting at [0, 0] and approaching the grammatical parse (Gr) at [1, 0]. Note the extra bowing toward the locally coherent attractor (LC) for *tossed*, causing extra slowing compared to *thrown*.

We simulated both conditions 2000 times using Euler forward discretization with a time step of 0.01, $D = 0.001$, and $\gamma = 0.25$. The system ran until it got within a small radius of an attractor. The local harmony h_1 of the locally coherent attractor ([0, 1]) was set to 0.8 in the *tossed* condition, and in the *thrown* condition to 0.5. As predicted, the system settled to the ungrammatical attractor in both cases, and it did so more frequently in the *tossed* condition (about 14% of runs) than in the *thrown* condition (<1% of runs). This increased the average settling time for *tossed* ($M = 159.073$ time steps, $SD = 27.692$) more than for *thrown* ($M = 149.794, SD = 24.698$), modeling Tabor et al. (2004)'s effect.

These simulations show local coherence effects for one parameter setting, but Fig. 4 shows how the same pattern holds

over a wide range of parameter settings. Where it does not hold, there is possibly empirical evidence for a phenomenon that corresponds to the model, different from local coherence. Fig. 4 shows mean settling times as a function of the harmony h_1 of the ungrammatical parse. We used $\gamma = 0.25$ here, but the pattern holds for a wide range of γ values. This figure shows that we will observe local coherence effects as long as $0 <= h_{1,thrown} < h_{1,tossed} < 0.85$. This predicts that local coherence effects should be widespread, a result supported by a large-scale eye-tracking corpus study (Bicknell et al., 2009).

For h_1 greater than about 0.85, the pattern changes: As the ungrammatical parse increases in harmony, the time its settling time approaches that of the grammatical parse, so it no longer pushes the overall average settling time up as much and the average settling time starts to drop (Fig. 4, bottom panel). The competition still causes a slowdown, but not as strongly as for somewhat lower-harmony competitors. Thus, the model predicts the strongest competition-induced slowdowns when the competing structure is of moderate harmony and smaller-magnitude slowdowns for both very low harmony competitors and (to a lesser extent) higher harmony competitors. This is, to our knowledge, unique among models of sentence processing. We speculate that this property of SOSP-TH might provide a new explanation for *ambiguity advantage* effects (e.g. Traxler, Pickering, & Clifton, 1998), where certain ambiguous relative clause and adjunct attachments are read more quickly than comparable unambiguous structures. If the harmonies of the two competing parses are close to 1.0 in the ambiguous condition but one is appreciably less than 1.0 in the unambiguous conditions, the competition-based SOSP-TH might be able to explain this puzzling effect that has been argued to rule out competition-based theories.

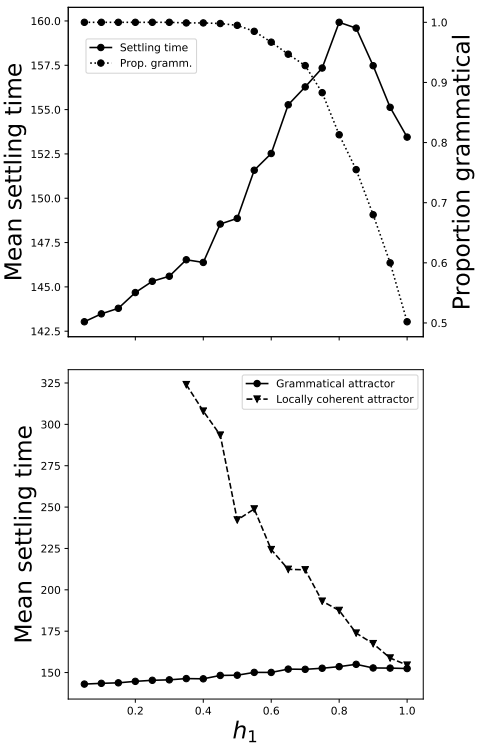


Figure 4: Top: Mean settling times for the local coherence model as a function of the ungrammatical parse h_1 (solid line, left y-axis) and the proportion of runs in which the the grammatical parse was selected (dotted line, right y-axis). Bottom: Mean settling time by selected parse (solid line, circles = grammatical; dashed line, triangles = locally coherent parse). For $h_1 < 0.4$, the system never settled on the ungrammatical attractor. Note the different y-axis ranges.

Discussion

In this paper, we presented a theory of timing effects in a self-organizing sentence processing (SOSP) framework, demonstrated how it can explain local coherence effects, and speculated on a possible new approach to ambiguity advantage effects. In our SOSP-TH framework, the amount of time it takes to build a structure depends on how well-formed the structure is, and the average structure-building time over many trials is the weighted average of settling times to each parse chosen.[6]

The local coherence model highlights the crucial role that lower-harmony structures play in SOSP-TH: A relatively well-formed but ungrammatical competitor slows processing more than a very ill-formed competitor because the higher-harmony competitor is built more often. This account differs from the grammar-supervised noisy channel approach to local coherence (Levy et al., 2009), which explains some (but not all; Kukona et al., 2014; Paape & Vasishth, 2015) local coherence effects by allowing the parser to edit its input to preserve grammaticality. By comparison, ACT-R for sentence processing (Lewis & Vasishth, 2005) might be thought of as partially grammar-supervised: Ungrammatical structures can affect processing via noisy memory retrieval that sometimes retrieves incorrect structures, but the cues used for retrieval are set by the grammar, preventing it from explaining local coherence effects via incorrect retrieval. By allowing both grammatical and ungrammatical structures to always influence processing, SOSP-TH occupies a unique and parsimonious place among theories of sentence processing.

[6]This is similar to recent cue-based retrieval approaches (e.g., Lewis & Vasishth, 2005) that model reading times with statistical hierarchical mixture models (e.g., Nicenboim & Vasishth, 2018).

Acknowledgments

This project was supported in part by NSF IGERT grant DGE-1144399.

References

Bicknell, K., Levy, R., & Demberg, V. (2009). Correcting the incorrect: Local coherence effects modeled with prior belief update. In *Proceedings of the 35th annual meeting of the Berkeley Linguistics Society* (pp. 13–24).

Cho, P. W., Goldrick, M., Lewis, R. L., & Smolensky, P. (2018). Dynamic encoding of structural uncertainty in gradient symbols. In *Proceedings of the 8th workshop on cognitive modeling and computational linguistics.*

Cho, P. W., Goldrick, M., & Smolensky, P. (2017). Incremental parsing in a continuous dynamical system: Sentence procesing in Gradient Symbolic Computation. *Linguistics Vanguard*.

Cho, P. W., & Smolensky, P. (2016). Bifurcation analysis of a gradient symbolic computation model of incremental processing. In A. Papafragou, D. Grodner, D. Mirman, & J. C. Trueswell (Eds.), *Proceedings of the 38th annual conference of the cognititve science society* (pp. 1487–1492).

Gerth, S., & beim Graben, P. (2009). Unifying syntactic theory and sentence processing difficulty through a connectionist minimalist parser. *Cognitive neurodynamics*, *3*(4), 297–316.

Haken, H. (1983). *Synergetics: An introduction* (3rd ed.). Springer-Verlag.

Hale, J. T. (2001). A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American chapter of the Association for Computational Linguistics on language technologies* (pp. 1–8). Association for Computational Linguistics. doi: 10.3115/1073336.1073357

Han, J. Y., Sayeh, M. R., & Zhang, J. (1989). Convergence and limit points of neural network and its application to pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, *19*(5), 1217–1222.

Kempen, G., & Vosse, T. (1989). Incremental syntactic tree formation in human sentence processing: A cognitive architecture based on activation decay and simulated annealing. *Connection Science*, *1*(3), 273–290.

Konieczny, L., Müller, D., Hachmann, W., Schwarzkopf, S., & Wolfer, S. (2009). Local syntactic coherence interpretation. evidence from a visual world study. In *Proceedings of the 31st annual conference of the Cognitive Science Society.*

Kukona, A., Cho, P. W., Magnuson, J. S., & Tabor, W. (2014). Lexical interference effects in sentence processing: Evidence from the visual world paradigm and self-organizing models. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *40*(2), 326–347. doi: 10.1037/a0034903

Levy, R. (2008). Expectation-based syntactic comprehension. *Cognition*, *106*(3), 1126–1177.

Levy, R., Bicknell, K., Slattery, T., & Rayner, K. (2009). Eye movement evidence that readers maintain and act on uncertainty about past linguistic input. *Proceedings of the National Academy of Sciences*, *106*(50), 21086–21090.

Lewis, R. L., & Vasishth, S. (2005). An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, *29*, 375–419.

McClelland, J. L., & Elman, J. L. (1986). The TRACE model of speech perception. *Cognitive psychology*, *18*, 1–86.

McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., ... Lee, J. (2013). Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st annual meeting of the association for computational linguistics* (pp. 92–97).

Muezzinoglu, M. K., & Zurada, J. M. (2006). RBF-based neurodynamic nearest neighbor classification in real pattern space. *Pattern Recognition*, *39*, 747–760.

Nicenboim, B., & Vasishth, S. (2018). Models of retrieval in sentence comprehension: A computational evaluation using bayesian hierarchical modeling. *Journal of Memory and Language*, *99*, 1–34.

Paape, D., & Vasishth, S. (2015). Local coherence and preemptive digging-in effects in German. *Language and Speech*, 1–17. doi: 10.1177/0023830915608410

Smith, G., Franck, J., & Tabor, W. (2018). A self-organizing approach to subject-verb number agreement. *Cognitive Science*. doi: 10.1111/cogs.12591

Smith, N. J., & Levy, R. (2013). The effect of word predictability on reading time is logarithmic. *Cognition*, *128*, 302–319. doi: 10.1016/j.cognition.2013.02.013

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. I: Foundations, pp. 194–281). MIT Press.

Stevenson, S. (1994). Competition and recency in a hybrid network model of syntactic disambiguation. *Journal of psycholinguistic research*, *23*(4), 295–322.

Strogatz, S. H. (1994). *Nonlinear dynamics and chaos: With applications to physics, biology and chemistry*. Addison-Wesley.

Tabor, W., Galantucci, B., & Richardson, D. (2004). Effects of merely local syntactic coherence on sentence processing. *Journal of Memory and Language*, *50*(4), 355–370.

Tabor, W., & Hutchins, S. (2004). Evidence for self-organized sentence processing: digging-in effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *30*(2), 431.

Traxler, M. J., Pickering, M. J., & Clifton, C. J. (1998). Adjunct attachment is not a form of lexical ambiguity resolution. *Journal of Memory and Language*, *39*, 558–592.

van der Velde, F., & de Kamps, M. (2006). Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, *29*, 37–108.

Vosse, T., & Kempen, G. (2000). Syntactic structure assembly in human parsing: a computational model based on competitive inhibition and a lexicalist grammar. *Cognition*, *75*, 105–143.

Vosse, T., & Kempen, G. (2009). The Unification Space implemented as a localist neural net: predictions and error-tolerance in a constraint-based parser. *Cognitive neurodynamics*, *3*(4), 331-346.

Explaining Decisions of a Deep Reinforcement Learner with a Cognitive Architecture

Sterling Somers (sterling@sterlingsomers.com)
Constantinos Mitsopoulos, Christian Lebiere ([cmitsopoulos, cl]@cmu.edu)
Department of Psychology, Carnegie Mellon University, 5000 Forbes Ave
Pittsburgh, PA 15213 USA

Robert Thomson (robert.thomson@usma.edu)
Army Cyber Institute, United States Military Academy, 2101 New South Post Road
West Point, NY, 10996 USA

Abstract

The work presented is an evaluation of a method for developing a hybrid system, consisting of a Deep Reinforcement Learning (RL) agent and a cognitive model, capable of providing explanations of its action decisions. The methodology uses a symbolic/sub-symbolic cognitive architecture to introspect on the activity of the network to understand its representation. The entropy in the system's behavioral predictions could be used as a signal to affirm or deny ascribing a representation to the network.

Keywords: deep reinforcement learning; cognitive modeling; introspection

Introduction

Deep Neural Networks (DNNs) have demonstrated an increasing success in various scenarios where they are used as function approximators. Their success is based in passing low-level sensory information through their structure, and creating higher-level abstractions of that information. The output of the network can be used either to classify inputs, predict, or in the case of RL, which is a learning paradigm for decision-making processes, to make decisions. As DNNs proliferate in both academic and private sectors, there will be an increased value to developing Deep RL agents that can be introspected upon. Being able to understand the concepts abstracted by the DNNs and how those concepts are factored into action decisions will allow us to develop specialized agents trusted to achieve their purpose.

The aim of this work is to develop a methodology for providing symbolic-level explanations for the action decisions of a Deep Reinforcement Learning (RL) agent. Our approach uses a cognitive architecture to model the internal representations of an RL agent at a symbolic level. In the long-term we envision a system that can inform a human observer why the RL agent made the decisions it made. This is particularly important whenever there is a mismatch between the knowledge of the observer and the knowledge of the RL agent. A mismatch could occur: a) when the RL agent produces an optimal but atypical solution, or b) when the RL agent fails to find a solution known to a subject matter expert (SME), or c) when the feature and action space are too complex for an SME to determine efficiently why a RL agent has made certain decisions. We envision that in a) and c), our system would be able to identify the most salient features of the environment attended by the RL agent, thereby explaining what features it is responding to. In b) our system would be able to identify representations available to an SME that are not shared by the RL agent. In this work we attempt to address b) by outputting symbolic state/action combinations consistent with the terminology used by an SME.

We believe that such a system should be generative and/or adaptive. While state and action category names should come from the SME, the system should be able to learn the mapping between states and actions for some subset of the RL agent's behavior.

The explanations produced by the system should also be constrained by human cognitive capacities. A deep RL agent is not limited with respect to the number of features it can attend to. An explanation system that fails to reduce the cognitive load on the observer is not useful as an explanation because the onus of determining the most salient features remains with the human observer.

In this work we evaluate a methodology where we create an adaptive cognitive model of an RL agent trained to play a simple mission in the real-time strategy game, StarCraft 2 (SC2). Our approach involves introspecting into the RL agent by mapping the activity of neural layers, as it makes action decisions, to symbolic output that can be used to generate explanations. This approach is reminiscent of Vinokurov et al. (2011), where they combined the hybrid cognitive architecture, ACT-R, with the neural-network-based architecture, Leabra (O'Reilly & Munakata, 2000) in an image classification task. We believe the computational cognitive architecture, ACT-R, is well suited for such a task because it is constrained by cognitive capacities that, in turn, will constrain explanations; it creates a symbolic model trace that can be tractably transformed into explanations; and it operates on sub-symbolic equations that are compatible with distributed representations in DNNs.

The cognitive model is initialized with knowledge consistent with an SME and evaluated against an RL agent with a sub-optimal policy. A sub-optimal policy represents knowledge mismatch situation b), described above, where the RL agent has failed to find a solution readily available to an SME. Although it may be possible to create an RL agent that performs the task optimally, our aim is primarily focused on explanation. Our explanation system is tasked with identify-

ing and reporting to a human user when the representations provided by the SME are not appropriate for describing the internal representations used by the network and to provide instead a representation that better describes the RL agent's internal state.

Our cognitive model attempts to adapt its internal representations to better match the action decisions of the RL agent. The cognitive model provides a model trace that can be used to infer an explanation. We use this trace in this investigation to determine if the RL agent has made an inference consistent with an inference made by an SME. High entropy in the model, in categories related to the inference, indicate that the RL does not make the inference. We confirm the accuracy of the model trace using t-SNE (Maaten & Hinton, 2008) to cluster the activity of the network. The t-SNE clustering confirms that the network makes three distinct abstract representations from the input space, which correspond to three state/action categories instead of the four that would result from making the same inference as the SME.

Task and Environment

StarCraft 2 is a real-time strategy game where players control units from a third-person perspective with the aim of eliminating opponents. In the screen-shot depicted in Figure 1, a military unit is selected and two beacons are present on the map.

Figure 1: Screenshot of a StarCraft 2 beacon mission.

SC2 provides a rich and complex environment for experimenting on various types of agents. The specific domain presents multiple challenges: imperfect information, macro and micro management of resources, strategic acting and multi-agent interactions. In addition, the SC2 Editor provides complete freedom to the mission's design, although anything beyond very basic missions provides a significant learning challenge for Deep RL agents (Vinyals et al., 2017).

The mini game we present here is straight forward. The goal of the game is to get the agent to one of two beacons: either the green beacon or the orange beacon. The green beacon presents a low-value target while the orange beacon represents a high-value target.

Despite the simplicity of the mission, the network learned a sub-optimal strategy. The agent clicks the highest rewarding beacon presented in every case. This strategy is successful in every case but one: where the green beacon is in the direct path between the agent and the orange beacon (blocking scenario). Clicking directly on the highest rewarding beacon makes the agent move in a direct line between its location and the beacon. In the blocking scenario, the agent moves towards the orange beacon, hits the green beacon, receives the low-value reward, and the game is reset.

Deep Reinforcement Learning Agent

The core of the agent consists of a deep neural network that attempts to solve a Markov Decision Process (MDP) defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R})$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ a finite set of actions, $\mathcal{T}$ is the transition probability for arriving in state s' when executing action a from state s, $\mathcal{R}$ is the reward function that defines the reward received for performing the aforementioned transition, and γ a reward discount factor. The goal of the agent is to maximize the expected return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ from each state s_t. The solution of the MDP consists of a function, named policy $\pi(.|s_t)$, that maps a state s_t to a distribution over actions that lead the agent to higher sums of rewards. The probability of performing action a_t in state s_t is denoted as $\pi(a_t|s_t)$.

The network parametrizes the policy π_θ with parameters θ. In this paper, we utilize the Advantage Actor Critic (A2C) algorithm which is the synchronous version of the A3C (Mnih et al., 2016). We adopt the same architecture (Figure 2) and implementation details as in Vinyals et al. (2017).

We consider the standard interaction between agent and environment. At each time step t the agent receives an observation s_t from the Starcraft II API and selects an action a_t according to its policy $\pi(.|s_t)$:

- **Observations:** consist of a set of image-like feature layers that represent the existence of a specific feature at a specific location on a screen. For example, in a 32×32 resolution map a feature that corresponds to enemy units type will be represented as a matrix with the same dimensions as the map. This feature matrix will have zero values apart from the elements that correspond to the pixels that are occupied by the enemy units. The value of these elements will be equal to the unit type identification number provided by the API. Generally, there are three main types of features: *map features* (entire map), *screen features* (part of map) and non-pixel information (e.g. player resources).

- **Actions:** There are two action categories: the action type and the action arguments. For example *click on* (action type), and x, y *location* (action arguments). In our examples, the action arguments will be the spatial location for performing the particular action type.

We trained the agent in the beacon task until it reached a steady performance relative to a human optimal score. We observed that the agent clicked on the highest value target in all cases. The RL agent exhibited no avoidance behavior in the blocking scenario.

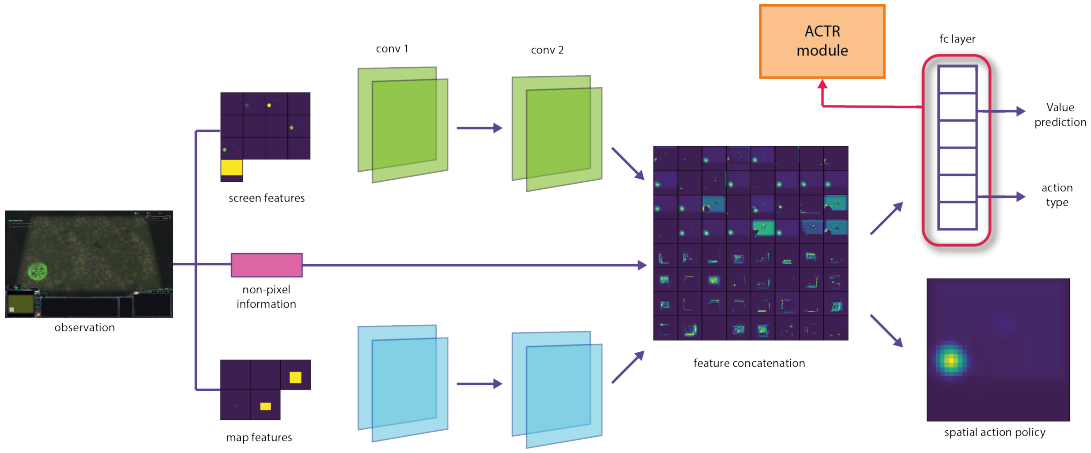


Figure 2: The Hybrid architecture: From observations, image-like features are generated for screen and map information. These are passed through two convolution layers and are concatenated with the non-spatial features. The value prediction, which represents the expected reward from the current observation, and the action type are determined by the concatenated feature representation passed through a fully connected (fc) layer with 256 units. The activities of this layer are sent to the ACT-R module for further processing. The spatial action is sampled from the probability distribution formed by a 1×1 convolved representation of the feature concatenation.

Cognitive Model

Our model is introspective in that it uses the activity of the network as the basis for ascribing representations. The RL agent and the model share a common ground: both play the game at the same time, the RL agent receiving numerical values describing the ground truth of the game state, while the model receives a symbolic version. Symbolic content comes from both interpreting signals from the game (presence of beacons) and an ontology created by SMEs. The ontology represents the kind of knowledge that a competent player should have. In the missions presented in this paper, the ontology is sparse: click on the green-beacon in green-only scenarios, click on the orange in orange-only scenarios, click on the orange in non-blocking scenarios, and go around the green beacon in blocking scenarios. Our cognitive model is developed in the ACT-R cognitive architecture.

ACT-R

ACT-R is a computational implementation of a unified theory of cognition (Anderson et al., 2004). It accounts for information processing in the mind via task-invariant mechanisms constrained by the biological limitations of the brain (see Anderson 2007 for an overview). The ACT-R architecture is organized as a set of modules, each devoted to processing a particular kind of information, which are integrated and coordinated through a centralized production system module.

The declarative memory (DM) and production system modules, respectively, store and retrieve information that corresponds to declarative knowledge and procedural knowledge. Declarative knowledge is knowledge that a person can attend to, reflect upon, and usually articulate in some way. Procedural knowledge consists of the skills we display in our behavior, generally without conscious awareness. Declarative knowledge in ACT-R is represented formally in terms of chunks. The information in the declarative memory module corresponds to personal episodic and semantic knowledge that promotes long-term coherence in behavior. In this sense, a chunk is like a data frame, integrating information available in a common context at a particular point in time in a single representational structure.

Each chunk has a base-level activation that reflects its past recency and frequency of occurrence. When a retrieval request is made, the most active matching chunk is returned from long-term declarative memory by an activation process. This process is computed as the sum of base-level activation, spreading activation, mismatch penalty and stochastic noise. Activation spreads from the current focus of attention, including goals, through associations among chunks in declarative memory. These associations are built up from experience, and they reflect how chunks co-occur in cognitive processing. The spread of activation from one cognitive structure to another is determined by weighting values on the associations among chunks.

Chunks are also compared to the desired retrieval pattern using a partial matching mechanism that subtracts from the activation of a chunk its degree of mismatch to the desired pattern, additively for each component of the pattern and corresponding chunk value. Finally, noise is added to chunk activations to make retrieval probabilistic, governed by a Boltzmann distribution.

While the most active chunk is usually retrieved, a blending process (Lebiere, 1999) can also be applied that returns a derived output reflecting the similarity between the values of the content of all chunks, weighted by their retrieval probabil-

ities reflecting their activations and partial-matching scores.

The flow of information is controlled in ACT-R by a production system, which operates on the contents of the buffers. Each production consists of if-then condition-action pairs. Conditions are typically criteria for buffer matches, while the actions are typically changes to the contents of buffers that might trigger operations in the associated modules. The production with the highest utility is selected to fire from among the eligible productions.

Mental Model

Our model is instance-based (Gonzalez et al., 2003). Whenever the model makes an action decision, it does so based on the similarity of the current situation to situations that it has stored in declarative memory. Instances are represented in the model as a chunk with the following five slots: *green*, *orange*, *blocking*, *vector*, and *action*. The slots green, orange, and blocking are binary True/False, the vector slot is populated with a chunk holding a 256-dimension vector, representing activity in the network (described below), and values for the action slot can be either: *select-green*, *select-orange*, or *select-around*.

Unlike many previous instance-based models in ACT-R, this instance-based model does not use ACT-R's blending mechanism since the output action is categorical in nature. Instead of blending a value estimate of the action category, our model simply selects the action from the stored instance that most closely matches the network activity of the RL agent.

Instances

We populated the model instances by tracing the RL agent while it plays different scenarios. In total, the RL agent played 50 games. Each game consists of a two-minute time period where a configuration of beacons is presented. The number of configurations presented varies based on the time it takes for the RL agent to reach a beacon. Once the agent reaches a beacon, a new configuration is presented, until the two minutes elapsed. Symbolic terms describing the scenario were stored in the instance (green-only, orange-only, green-and-orange). We populated the *action* slot of the model's declarative memory with the action an expert would choose: select-green in a green-only scenario, select-orange in both orange-only and non-blocking scenarios, and select-around (as an abstract action) in the blocking scenario. Importantly, given these instances, the model would never predict that the RL agent would select the orange beacon in a blocking scenario. Finally, stored for every instance in declarative memory, was a chunk containing a vector representing the network activity at that instant of game-play (introspection).

Introspection We augmented the symbolic context representation in the instance chunks with a vector that represents the network activity from the *fc* layer (top right, Figure 2). We chose this layer for two reasons. First, we wanted a layer that would be abstract enough to represent scenarios (as opposed to features in the scenarios). Second, we wanted a layer that was spatially invariant. The *fc* layer corresponds to the non-spatial action selection and chooses which action type to take but does not choose the screen location of that action to be executed. For example, it might be responsible for choosing a click-action but is not responsible for choosing where to click.

For efficiency, we follow Sanner et al. (2000), and limit the number of instances that are stored. In our case, new instances are limited to 10 items per category such that the distance of the new vector must be greater than the minimum distance between vectors across all categories. After initial testing, we limited the total number of instances per category to ten. Since each instance also stores a vector of network activity and the vectors drive the similarity measure in ACT-R, we attempted to capture a large space of vectors in each category. We chose to filter the creation of categories by maximizing the euclidean distance between vectors.

Instance-Based Learning and Instance Retrieval

Once we populated the initial set of instances in ACT-R's declarative memory, it tries to classify new game states. We use ACT-R's partial matching mechanism such that it tries to retrieve from memory instances that are most similar based upon a vector representing network activity. The similarity metric we used was euclidean distance between vectors, normalized to a scale of 0 to -1, for all vectors in the category. All other slots are not matched upon. The result of this process is that the instance with the closest vector is recalled, regardless of the symbolic content of that memory. It is possible, therefore, that symbolic content of the recalled instance does not match the symbolic content of the current game state. In fact, we are expecting that in blocking cases, the model will retrieve a non-blocking category. Once an instance is retrieved, a production that matches the action stored in the instance will fire. The production that fires is recorded for evaluation. A side effect of the production is to combine the data from the current game state and the instance retrieved from memory to create a new memory.

Memory Creation for Model Adaptation The process of creating new memories is the central feature that makes the model adaptive. As we allow for a mismatch between the symbolic content of the game state and the symbolic content of the recalled instance, the model is capable of creating new state/action categories. The new state/action categories can then be used by the model as it continues to play the game. The new memories store the vector for the current activity of the RL agent network, the current game state (green, orange, blocking), and the action from the retrieved instance, combined as a new chunk.

Evaluation

The goal of our system is to provide symbolic terms, and a model trace that can be used to infer an explanation. Our models introspects on the network and ascribes plausible representations to it.

Although it is typical to assess a model on how well it predicts behavior, our interest is not how well the model fits the network but what the performance might tell us about the appropriateness of an ascribed mental state or explanation. Since we conducted an evaluation of the network's competency, we know ahead of time that the network does not appear to represent a spatial relationship between the agent and the beacons. We inferred from this behavior that the network does not make a spatial inference. We hypothesized that by using partial matching on the vector slot only, the model would conflate blocking and non-blocking scenarios, occasionally retrieving the select-orange action (appropriate in non-blocking scenarios) and create a new instance category: green, orange, blocking, select-orange.

Analysis

Since each prediction that the model makes is categorical (an action), we treat each guess as binary (correct/incorrect).

We ran the model 100 times. Each run consisted of 10 episodes of the game, and each episode consists of roughly 25 randomly generated scenarios: green-only, orange-only, green-and-orange-non-blocking, and green-and-orange-blocking. It was difficult to estimate a number of simulation runs that would be practical and provide a reasonable estimation of the model's performance. Variance was observed to be stable at 100 runs. Importantly, since we are not overly concerned about model fit, finding a true mean was not absolutely essential.

Percent Correct The model correctly predicted the actions of the RL 70.09 % of the time across all runs. In green-only scenarios, the model predicts 99.95 % of the actions. In orange-only scenarios, the model predicts 89.39 % of the actions. In green-and-orange (Blocking and Non-Blocking) scenarios, the model predicts 44.85 % of the actions. This can be broken down further into two sub-categories: blocking and non-blocking scenarios. In blocking scenarios, the model predicts 36.04 % of the actions. In non-blocking scenarios, the model predicts 49.04 % of the actions.

New Categories As previously mentioned, the model has knowledge of what action to choose in a green-only scenario, an orange-only scenario, non-blocking scenarios, and blocking scenarios. The model attempts to make new categories based upon the RL network activity.

Table 1 summarizes what scenario/action pairs were used and created by the model. The top of the table are the state/action categories that the model was initialized with and the bottom of the table are the state/action categories created by the model, during game play. The left most column is the scenario categories. Column 2 (Action) is the associated action taken by the network in that scenario. Column 3 (Cat./Run) represents the average frequency the scenario/action category was created per model run. Column 4 is measure of how often the associated action was chosen in the given scenario. Note that the symbolic content in the table (e.g. *blocking, select orange*) are the categorical output of the cognitive model, that can be used to infer explanations.

Table 1: Category Creation and Use

Scenario	Action	Cat./Run	Chosen (%)
Green-Only	Select-Green	default	99.95
Orange-Only	Select-Orange	default	88.39
Non-Blocking	Select-Orange	default	49.04
Blocking	Select-Around	default	63.88
Blocking*	Select-Orange	9.98	36.04
Green-Only	Select-Orange	<0.01	0.03
Orange-Only	Select-Green	3.03	3.63
Non-Blocking	Select-Green	0.40	0.18
Blocking	Select-Green	0.30	0.17
Green-Only	Select-Around	0.01	0.02
Orange-Only	Select-Around	5.48	7.98
Non-Blocking*	Select-Around	10.00	50.80

* indicates expected categories.

Clustering We performed t-SNE clustering (Maaten & Hinton, 2008) on the activity of the *fc* layer of the RL network in order to investigate the network's representations. The results of the clustering are illustrated in Figure 3. As illustrated, the network has distinct categories for both the green-only and orange-only scenarios. Also illustrated in the figure, green-and-orange (non-blocking) and blocking scenarios show a complete overlap.

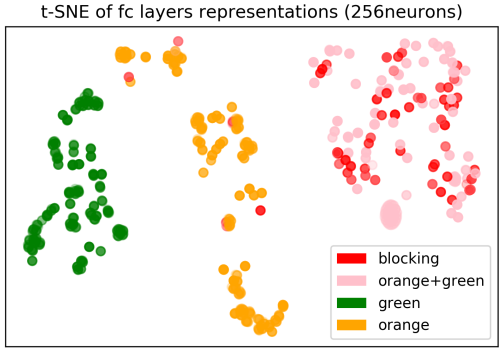


Figure 3: The RL network's representations formed at the fully connected layer are naturally clustered depending on the scenario that the agent is facing. The t-SNE process identifies 3 clusters, coloring is the result of semantic labelling.

Discussion

We know from evaluating the RL agent that it does not perform an action consistent with Select-Around. There are two ways to interpret this behavior: first, that the RL agent does not know how to go around; or second, that the RL agent does not have the concept, 'blocking', and therefore does not distinguish blocking scenarios from non-blocking scenarios.

The results of our model suggest that the latter is the case. Our model attempts to use the concept 'blocking' and creates a new state/action category: Blocking/Select-Orange, consistent with the behavior of the RL agent. Although this state/action category reflects the action of the agent, the cognitive model only approaches 50 % success in its predictions. This is a first indication in the trace that the model does not create the 'blocking' category (as opposed to an inability to go around).

Although we had not originally expected it, the cognitive model creates the state/action category Non-Blocking/Select-Around. The creation (and high usage) of the Non-Blocking/Select-Around in combination with the Blocking/Select-Orange category suggests that the RL agent cannot distinguish between green-and-orange-non-blocking and green-and-orange-blocking: that is, it suggests the RL agent does not have the category 'blocking'.

These findings are confirmed in the t-SNE clustering. In particular, the overlap between the pink and red dots in Figure 3 suggest that RL agent conflates blocking and non-blocking categories. Just as in the model, the clustering algorithm reveals only three main categories: green-only, orange-only, and green-and-orange.

Other categories created by the cognitive model (e.g. Orange-Only/Select-Around), although unexpected, are likely due to noise in either our similarity measure (not sensitive enough) or the RL's distributed representation. For the most part, those categories have both a low category creation rate and low usage percentage. It is worth pointing out that the Orange-Only/Select-Green and Orange-Only/Select-Around have a high-category creation per run (3.03, 5.48 respectively) and percent chosen (3.63 percent and 7.98 percent of the time) compared to the other unexpected categories. Overall, their usage remains quite low.

Conclusion

The work presented in this paper is an initial evaluation of a methodology for generating explanations for the behavior of Deep Reinforcement Learners. We use a computational cognitive model to introspect upon the activity of the network. Given an initial set of classifications, defined by categories present in an ontology, the model either uses existing categories to choose an action (a prediction of the network's action) or creates a new instance by combining content from the current game-state and the retrieved declarative memory instance.

We believe the output of the model can be used to generate explanations and, in particular, our system is able to detect when concepts from the ontology (the concepts humans use) are not realized in the RL agent. The model presented only begins to scratch the surface of what can be accomplished in the overall methodology.

We are currently pursuing three lines of research to expand our approach. We have developed a method for leveraging the blending mechanism in ACT-R to determine which features are most salient in an action decision. Providing the key features that were used in an action decision helps make the explanation more tractable for a naive observer. We are developing top-down interaction such that the ACT-R model can influence the training of the Deep RL agent. We do this to try and influence the agent to learn a concept that it has had trouble learning or that appears in the ontology but not the RL agent. There may be times when we want a more natural mapping between a human observer and an RL agent so that explanations are more straightforward. Finally, we are in the process of developing an ACT-R model that learns the task in a human-constrained manner.

Acknowledgments

This work was funded by a subcontract from PARC under DARPA contract FA8650-17-C-7710.

References

Anderson, J. R. (2007). *How Can The Human Mind Occur In The Physical Universe?* New York, NY: Oxford University Press.

Anderson, J. R., Bothell, D. J., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004, oct). An integrated theory of the mind. *Psychological review*, *111*(4), 1036–60. doi: 10.1037/0033-295X.111.4.1036

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

Lebiere, C. (1999). The dynamics of cognition: An ACT-R model of cognitive arithmetic. *Kognitionswissenschaft*, *8*(1), 5–19. doi: 10.1007/s001970050071

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).

O'Reilly, R. C., & Munakata, Y. (2000). *Computational Explorations in Cognitive Neuroscience* (Vol. 46). MIT Press.

Sanner, S., Andrew, S., Edu, C. M. U., Anderson, J. R., Lebiere, C., Andrew, C. L., ... Lovett, M. (2000). Achieving Efficient and Cognitively Plausible Learning in Backgammon. In *Seventeenth international conference on machine learning* (pp. 823–830). Stanford, California.

Vinokurov, Y., Lebiere, C., Herd, S., & O'Reilly, R. (2011). A Metacognitive Classifier Using a Hybrid ACT-R/Leabra Architecture. In *Proceedings of the 15th aaai conference on lifelong learning* (pp. 50–55).

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., ... others (2017). Starcraft ii: a new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

Modeling prototype effects in a binary classification task

Robert St. Amant (robert.a.stamant2.civ@mail.mil),
MaryAnne Fields (mary.a.fields22.civ@mail.mil),
Craig Lennon (craig.t.lennon.civ@mail.mil)
U.S. Army Research Laboratory, 28000 Powder Mill Rd. Adelphi, MD

Abstract

This paper analyzes the results of two experiments in the early classification literature. A Naive Bayes model provides a good account of human performance, as expected, though category-level differences remain unresolved. An ACT-R cognitive model is described that reproduces some but not all of observed patterns in human performance; directions for further research are identified.

Keywords: Classification, categorization

Introduction

Categorization is a topic of longstanding interest in psychology, cognitive science, and artificial intelligence. Plato famously captured our fascination in his metaphor of "carving Nature at its joints"—how we divide things into categories can reveal important insights about how we conceive of the world. Our ability to categorize is plausibly a form of cognitive economy; if we can identify a stimulus with a known category, we can make inferences about information we cannot directly observe. Further, to the extent that our categories map to structure in the world, categories can help us identify the possible actions we can take (Rosch & Lloyd,1978).

Categories are a basic part of our rationality. Anderson has applied rational analysis to categorization, providing a Bayesian framework in which the incremental, iterative process of categorization that people follow can be understood (Anderson,1991). This work has led to computational models of categorization, such as the hybrid model of Anderson and Betz (2001), which included separate rule-based and exemplar-based strategies for categorization.

We have adopted the ACT-R cognitive architecture (Anderson,2009) for research on mobile robotics in our laboratory. We would like to take advantage of architectural capabilities that support categorization and classification, for such tasks as scene and object recognition, and even facial expression categorization (Fields, Lennon, Martin, & Lebiere,2017). Our interest is in problems that a robot encounters in making sense of patterns of features in its environment. For example, imagine an assemblage of a table and chairs: Is this a classroom? An office? A dining room? A public cafe? A judgment may depend on geometry and other factors, but we would also like to depend on a cognitive model's categorization as having the same basic properties as that of a human.

In a seminal paper, Rosch and Mervis (1975) elucidated some of those properties through a series of experiments. They established several findings: Categories have internal graded structure; items with a higher family resemblance within the category are judged as being being more representative than other items. The degree of prototypicality of an item in a category depends on other items in the category. Prototypicality of an item in one category can be induced by overlap with items in a contrasting category.

This paper revisits two of Rosch and Mervis's experiments in an attempt to explain their findings in terms of a unified cognitive architecture. One motivation for this paper is that Anderson's extensive work on applying rational analysis to categorization has not intersected only indirection with Rosch's work; we saw this as a gap we might fill. In the next section, we summarize Rosch and Mervis's experiments, including their results. The experimental tasks are amenable to modeling in part because they are based on artificial, constructed data. This means that contextual effects of prior knowledge were negligible (e.g., in comparison with natural categories), which significantly reduces the challenge of building a computational model. The section that follows gives a probabilistic analysis that accounts for most of the findings discussed. Computational modeling issues are then are addressed.

This paper makes a set of incremental contributions to the literature. First is a new analysis of historically important experiment results; this required problem formalization and some reconstruction of the original data. Second is a demonstration that Anderson's rational analysis applies well to these classification tasks, and that a specific Naive Bayes classification method predicts experiment observations. Third is the identification of challenges raised for modeling classification tasks, with suggested directions for future work.

Related Work

Probabilistic and information processing models of categorization and classification have become sophisticated, able to capture fine performance distinctions and to give insight into cognitive processing (Kruschke,2008;Pothos & Wills,2011). As suggested in the introduction, our interest in classification is relatively narrow, focusing on processes that are, ideally, consistent with the rational analysis and the ACT-R architecture.

The most obvious related work is Anderson (1991)'s rational analysis, plus follow-on work with ACT-R (e.g., Anderson & Matessa,1992;Petrov & Anderson,2000;Anderson & Betz,2001); this will figure in our discussion below. For now, we note that early work by Rosch and others tended to focus on classification tasks, where experiment participants are asked to assign items to pre-defined or well-understood existing categories. Categorization tasks now commonly allow

participants to form new categories to account for the items they see, but categorization is beyond the scope of this paper.

Two experiments described by Rosch and Mervis (1975) involve learning categories of artificial data. Briefly, their Experiment 5 had one control category and two contrast categories, as shown in Table 1.[1] The focus of this experiment was on *family resemblance*, here denoted R_c of an item in a category. An item consists of some number of distinct terms; e.g., KT3YR has five terms: K, T, 3, Y, and R. A term can occur in multiple items in a category; e.g., K occurs 3 times over all the control items (the first column of Table 1). R_c for a given item in category c is computed by summing the occurrences of each of its terms; e.g., $R_c = 3+3+3+1+2 = 12$ for the control category—K appears 3 times across all control items, T also 3 times, and so forth. The intuition behind this computation is that we consider items to be more similar to each other the greater the number of features, or terms, they share in common. Over a fixed set of items that form a category, the commonality of a term is equivalent to the count of its appearances.

Rosch and Mervis defined two contrast categories, one "symmetric" and the other "asymmetric". The symmetric category is so-called because it contains paired terms with low ($R_c = 15$), medium ($R_c = 19$), and high ($R_c = 21$) family resemblance values. An additional experiment factor could thus be used to test for differences within a family resemblance level. All R_c values for terms in the asymmetric category are distinct. The details of item construction for the different categories are not relevant here, but one ramification is: in the symmetric category the mean family resemblance of items is higher (and in the asymmetric category, lower) than that of the control category.

Variations on these items were presented to experiment participants: the characters in each string were shuffled (remaining constant afterwards), and they were presented in a randomized order. Each participant saw the control items and one set of contrast items, each with its association to its category. After the initial presentation, participants were tested on the items, each shown without its category, in random order; correct and incorrect answers were marked as such to the participants. Once the items had been learned, indicated by a successful run twice through all items in both categories, a testing phase collected response times for the items and other judgments about them.

Rosch and Mervis hypothesized that family resemblance of an item to its category would affect performance in classification. Higher family resemblance items would be classified with fewer *Errors* and a lower *Response Time*; experiment participants would also judge such items to have higher *Prototypicality* with respect to their category. These measures showed no significant differences between the items in the Control category, where resemblance was constant. With the categories for comparison divided into low, medium, and high resemblance items (those divisions shown as gray lines in Table 1), all of the predictions held.

For Experiment 6, Rosch and Mervis defined new categories for comparison, two of which are shown in Table 2. Note that Experiment 5 is a very easy binary categorization task: the categories being compared have no terms in common. Experiment 6 categories were different. The items in the contrast category were constructed so as to provide a different degree of *overlap* with the category for comparison. The overlap of an item with its contrast category is computed as the number of its items that appear in the contrast category. For example, JXPHM has zero overlap: none of its characters appear in any item in the contrast category. HMQBL has an overlap of 3, because Q, B, and L appear one or more times. Resemblance (R_c) and overlap (L_c) are shown for each item.

Rosch and Mervis hypothesized that, for the same category structures in Experiment 5, overlap with the contrast categories would produce different results: higher overlap would lead to greater error rates, longer response times, and lower judgments of prototypicality. In brief, overlap (or lack thereof) can induce prototypicality. All of these predictions held, for low, medium, and high overlap.

A Naive Bayes model

Naive Bayes classifiers have a long history in machine learning and information retrieval. In this section we outline a simple classifier and connect its structure with the data described in the previous section. This section follows the exposition and adapts the formal notation of Manning, Raghavan, and

[1] Ideally, the items in Table 1 would match those in Table 3 of Rosch and Mervis (1975), but the latter contain errors: the control group for the "nonoverlap contrast category" is described as containing items that all have a family resemblance of 12, but their actual values range from 11 to 14. We could not easily identify the problem and (after contacting the authors of the original paper) opted to construct new strings to reflect the relevant criteria.

Table 1: Experiment 5 categories

Control	R_c	*Symmetric*	R_c	*Asymmetric*	R_c
KT3YR	12	PXWLC	21	8LP4M	16
KT301	12	XWLCV	21	8LP4S	15
7U3Z1	12	SPXWL	19	8LPXW	14
HUBZ7	12	WLCV2	19	8LCV2	12
HUBQR	12	4SPXW	15	8DNGE	10
KTBQA	12	LCV28	15	M9N56	7

Table 2: Experiment 6 categories

Symmetric	R_c	L_c	*Contrast*	R_c	L_c
JXPHM	15	0	GVRTC	15	0
QBLFS	15	5	SFLBQ	15	5
XPHMQ	19	1	VRTCS	19	1
MQBLF	19	4	CSFLB	19	4
PHMQB	21	2	RTCSF	21	2
HMQBL	21	3	TCSFL	21	3

Schütze (2008). Our discussion is to some extent a reiteration of Anderson (1991)'s classic rational analysis of categorization, much simplified because it does not involve the formation of new categories.

In Rosch and Mervis's experiments we have a classification task, with a set of disjoint *categories* that partition a set of items. An *item* is a set consisting of a fixed number of *terms*. For example, as mentioned above, the first column in Table 1 shows a category, and each row in that column is a different item, a set of terms; the first is $\{\mathrm{K},\mathrm{T},3,\mathrm{Y},\mathrm{R}\}$. Every item x has a category c, a distinguished feature that we will express as $x_{cat} = c$. Because exactly two categories are involved, we use c to represent one category and $\bar{c}$ the other.

We assume a maximum *a posteriori* (MAP) decision rule, in which the category chosen for an item is the most probable.

$$P(c|x) = sP(c)\prod_{t\in x} P(t|c), \tag{1}$$

where $P(t|c)$ is the probability that some term t appears in an item in category c; s is a scaling factor. Note the independence assumption: the product stands in for $P(x|c)$, assuming conditional independence between the component terms of x given c.

We can estimate $P(c)$ in Equation 1 by counting:

$$\hat{P}(c) = \frac{|\{x : x_{cat} = c\}|}{|\{x : x_{cat} = c\}| + |\{x : x_{cat} = \bar{c}\}|}; \tag{2}$$

$P(t|c)$ can also be estimated in a straightforward way:

$$\hat{P}(t|c) = \frac{|\{x : t \in x\} \cap \{x : x_{cat} = c\}|}{|x|\,|c|}, \tag{3}$$

where the numerator is the count of items in category c that contain term t, and the deonominator is the product of the count of all items in c and the size of each item x, assumed constant. (For conciseness in later formulas, we will write $count(t,c)$ for the numerator.) For example, for the specific term K and the Control category of Experiment 5,

$$\hat{P}(\mathrm{K}|\mathrm{Control}) = \frac{|\{\mathrm{KT3YR},\mathrm{KT301},\mathrm{KTBQA}\}|}{5\cdot 6} = \frac{3}{30} = .10,$$

which can easily be verified in Table 1.

We expand Equation 1 as follows, using a Bayesian likelihood formulation and taking logarithms:

$$\log\left[\frac{sP(c)}{sP(\bar{c})}\frac{\prod_{t\in x}P(t|c)}{\prod_{t\in x}P(t|\bar{c})}\right] \tag{4a}$$

$$= \log\frac{P(c)}{P(\bar{c})} + \log\prod_{t\in x}\frac{P(t|c)}{P(t|\bar{c})} \tag{4b}$$

$$= \log\frac{P(c)}{P(\bar{c})} + \sum_{t\in x}\log P(t|c) - \sum_{t\in x}\log P(t|\bar{c}) \tag{4c}$$

ACT-R modelers will find this expansion familiar.[2] Anderson (1996) described 4b as the activation equation:

$$\log(\textit{posterior odds}) = \log(\textit{prior odds}) + \log(\textit{likelihood ratio}) \tag{5}$$

We will thus refer to Equation 4c as the "posterior odds" formula (or simply *odds*, in the caption of Figure 1). We can approximate it by substituting Equations 2 and 3 as estimates of $P(c)$ and $P(t|c)$.

Returning to Rosch and Mervis, family resemblance (R_c), for an item x and its category c, is the sum of the appearances of each term of x over all items in c. For overlap (L_c), each term in x is counted if it also appears in some item of $\bar{c}$ (with the minimum function ensuring that a term is counted only once, if it appears at all). R_c and L_c are plausible proxies for the second and third terms of Equation 4c, respectively.[3]

$$R_c(x,c) = \sum_{t\in x} count(t,c) \tag{6a}$$

$$L_c(x,c) = \sum_{t\in x} \min(count(t,\bar{c}),1) \tag{6b}$$

The posterior odds formula can be used to predict the Errors, Response Time, and Prototypicality measures of the previous section. Figure 1 shows a scatter plot for each measure. Data points are the low, medium, and high resemblance subsets of items in the Symmetric and Asymmetric categories from Experiment 5; the Control category items from Experiment 5; and the low, medium, and high overlap subsets in the Symmetric category from Experiment 6.[4] The horizontal axis in each plot gives the range of log posterior odds values, as produced by Equation 4c, using Equations 2 and 3 for estimation. The vertical axis shows the range of the relevant performance measure. We note a strong linearity in the Errors and Response Time plots. Linear least squares fits are given in the caption of Figure 1. Our interpretation is that the posterior odds formula provides a reasonable explanation for patterns in the data.

The subsets within the plots of Figure 1 are informative. Within each experiment condition, low/medium/high relationships are correctly predicted over all three measures. For

[2] Equation 4b could be simplified further (the first term drops out because $|c| = |\bar{c}|$) and approximated as $\log\prod_{t\in x}\frac{count(t,c)}{count(t,\bar{c})}$. We leave it in the form of Equation 4c to emphasize that the posterior odds are increased by family resemblance, decreased by overlap.

[3] As a philosophical point, our description of the direction of this relationship is intentional. The Bayesian account is not simply an approximation of what people do. Rather, following Anderson (1991)'s rational analysis of cognition, we would say that R_c and L_c influence human performance in categorization (to the extent that they do) *because* they contribute as approximations to rational behavior.

[4] Rosch and Mervis (1975, p. 593): "Contrast strings in control... were not analyzed in Experiment 6."

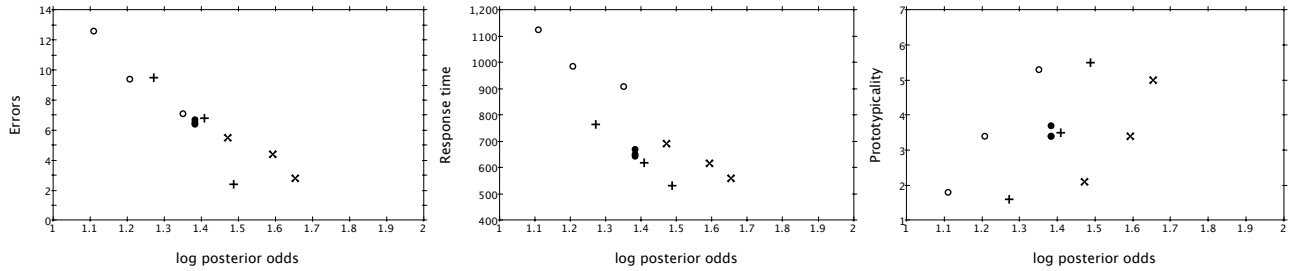


Figure 1: Prediction of errors, response time, and prototypicality ratings. Experiment 5: Control (•); Symmetric (×); Asymmetric (+); Experiment 6: Symmetric (○). $Errors = 31.27 - 17.66odds$, $R^2 = .877$; $Time = 2148.06 - 1017.96odds$, $R^2 = .720$; $Prototypicality = -2.52 + 4.33odds$, $n.s.$

example, for Response Time, the posterior odds formula predicts lower times for higher family resemblances; this holds for both the symmetric and asymmetric comparisons of Experiment 5. The effect of overlap in Experiment 6 is also predicted: although the family resemblance values are the same as for the symmetric category in Experiment 5, Experiment 6 performance is degraded significantly.

The model is too coarse to predict some inter-category relationships, however. Items with the same approximate family resemblance value give different performance, depending on the category. For example, in Experiment 5 the high $R_c = 15$ or 16 asymmetric items have a response time of 532 ms, while the $R_c = 15$ symmetric items have a much longer duration of 692 ms. This difference is systematic, with the pattern holding for all three measures. This appears to be due to the distribution of items (or rather, their constituent terms) within the categories: some distributional factor other than R_c or L_c allows items in the asymmetric category to be handled more efficiently.

Our analysis has obvious limitations, aside from using sparse data to develop and evaluate a model. Ordering relationships within an item, as well as an item's identity as a whole, are ignored. The sensitivity to possible interactions between individual items and other items in the category is also ignored.

These and other characteristics of our analysis are driven in part by our choice of a classification method. We have described a Naive Bayes classifier based on a multinomial event generation model (Manning et al.,2008;McCallum & Nigam,1998). The specifics of the domain allowed for several constraints that are not part of the more general multinomial model, as mentioned above. In addition, the multinomial model counts occurrences of terms in items, while in our classification problem terms can occur at most once in an item.

Consider the size of the aggregate vocabulary of the items to be classified, in terms of binary features. Each item has five explicit terms (e.g., $|\{\mathrm{K},\mathrm{T},3,\mathrm{Y},\mathrm{R}\}| = 5$), but the feature structure of the entire classification problem is much larger. The number of features for each pair of categories in the experiments is the size of the union of all items in both categories, $|\bigcup(\{x : x_{cat} = c\} \cup \{x : x_{cat} = \bar{c}\})|$, which ranges from 20 in Experiment 6 to 32 for the asymmetric comparison in Experiment 5. Given the response times in the neighborhood of 500 ms to 1200 ms, the explicit consideration of the absence of terms does not seem cognitively plausible.

In sum, this analysis provides evidence that multinomial Naive Bayes, a simple, well-understood model of classification, can produce quantitative predictions of the prototype effects identified by Rosch and Mervis: response time, error rates, and prototypicality judgments as influenced by family resemblance and overlap with contrast categories. This is a minor result, consistent with Anderson (1991)'s results, but to our knowledge this specific formalization and analysis are not part of the literature.

Cognitive modeling considerations

Given the analysis in the previous section, with the resulting statistical model, the next step is to build a cognitive model to carry out the classification tasks of Experiments 5 and 6. For pragmatic reasons, we focused our modeling efforts on explaining response time. We wanted to account for three phenomena in models that replicate human performance in Experiments 5 and 6:

- Family resemblance: Items with a higher R_c have lower response times.
- Overlap: Items with higher L_c have higher response times.
- Category-level differences: Response times depend on category properties; for example, items in an asymmetric category have a lower response time than items in a symmetric category, even if their R_c is the same.

We developed a modeling scenario as follows. In a training stage, items appear as strings of characters on the display, each with a known category. The model attends each item, breaking it down into component features in the imaginal buffer, with further processing to update declarative memory, as described below.[5] The follow-on testing stage is similar

[5] Also, as a "mechanical" extension to facilitate model development, we added the capability of treating the slots of a chunk as representing an unordered set of items (rather than as a relation with named places).

in structure, except that after the decomposition of the item, a memory retrieval is performed, with the item's features in the imaginal buffer to provide context: the category with the highest activation is retrieved. The model presses a key corresponding to the category of the item.

Rosch and Mervis's experiments provide a useful timing constraint on the details of this model; response times as short as 532 ms were observed for some experiment conditions. As a comparison, the visual and motor processing for an ACT-R model to acquire an item and press a key, using default parameter values, takes 360 ms.

One implication is for the decomposition of a visual item into its consitituent characters during the testing stage. If a model were to sequentially focus attention on each character (185 ms per shift), the process would take close to a second. Instead, we assume that the letters in each item are acquired in parallel (McClelland & Rumelhart,1981) with an imaginal buffer action, reusing parts of a model we developed for transcription typing (St. Amant, Goodwin, Dominguez, & Roberts,2015). The addition of this capability extends the response time in the model to 495 ms.

The more critical conceptual issue is about the activation of chunks representing categories, items, and terms in memory. Anderson (1996) explains how this works in ACT-R (the contemporary version), in the context of categorization:

> Various features in chunks spread activation to various categories according to the likelihood of that feature given the category. Categories have base-level activations to reflect their prior frequencies. The resulting activations reflect the log posterior probability of the category. The most active category is chosen to classify the stimulus.

Consider a snapshot in time during the testing stage. Chunks for both categories c and $\bar{c}$ are present in declarative memory, each with slots holding chunks that correspond to the terms that have appeared in items learned for the categories.[6] Each category chunk has been presented the same number of times, and though differences in rehearsals during the training period could produce different base activation levels, the simplest interpretation is that $P(c) = P(\bar{c})$. Differences in activation are thus produced by the posterior probability, the second term in Equation 4b.

Unfortunately, spreading activation for a category chunk c, which we will write S_c, doesn't quite capture the posterior probability, in ACT-R 7 (Bothell,n.d.).

$$S_c = \sum_{t \in x} W S_{tc} \tag{7a}$$

$$= \sum_{t \in x} W \left[S - \ln(fan_{tc}) \right] \tag{7b}$$

$$= \sum_{t \in x} W \left[S - \ln\left(1 + count(t,c) + count(t,\bar{c})\right) \right] \tag{7c}$$

Equation 7c suggests that spreading activation could account for the results of Experiment 6. For our purposes, the weight W is assumed constant. On retrieval of a category for a set of terms x, activation will be reduced by the fan, a function of how many chunks in declarative memory reference the terms in x. Because of overlap, the fan is greater than it would be otherwise. Equation 7c is problematic for family resemblance, however. Items with a high family resemblance in a category, by definition, have a greater number of terms that are contained in other items in the category. This reduces S_c. To the extent that this factor influences activation, Rosch and Mervis's results imply that the opposite should be the case. Finally, it is not obvious how the last phenomenon of category-level differences can be addressed.

We experimented with alternatives that extended the architecture to support the necessary computations in different ways. At the core of the extensions was handling co-occurrences of t and c during the training phase, to support computation of the posterior probability during the test phase. The models required careful tuning event to approximate the results of Experiment 5 and 6—and then only of the family resemblance and overlap phenomena, not the category-level differences. Excluding data from Experiment 6, Figure 2 shows the best model we were able to develop. The model's predictions, unfortunately, are not very good, with $RMSE = 77.80$, and a linear fit that is not significant.

One extension recorded frequencies explicitly, combining them into similarities between t and c chunks, to enable the partial matching mechanism to produce this computation. A different extension automatically "distributed" activation from an item chunk to its terms, during the training phase when the item was added or merged into declarative memory; reconstructing an appropriate activation during the test phase was problematic. Our efforts turned out to be mainly atheoretical software modifications that blurred the distinction

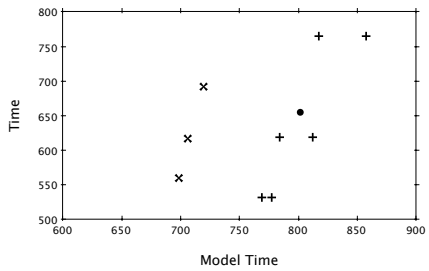


Figure 2: Model prediction of response time, Experiment 5: Control (•); Symmetric (×); Asymmetric (+).

[6] Given the conditions of Rosch and Mervis's experiment, it's instead possible that participants were recalling specific items, each with its category feature, rather than chunks representing the categories themselves. This does not change the analysis, however.

between the contributions to activation of prior and posterior probability.

More promising alternatives are in the literature (e.g., Van Maanen & Van Rijn,2006;Thomson, Harrison, Trafton, & Hiatt,2017). We have begun to explore new directions, in particular based on the work of Thomson et al. (2017), who propose an alternative for spreading activation. Their approach is based on Hebbian associative learning, in which the strength of associations between items in memory changes over time due to temporal proximity. We have not yet built new models, however.

The phenomenon of category-level differences also remains to be addressed. Our characterization of the phenomenon at the category level implies that properties of a category (which is defined only by the presentation of its items) influence its activation and thus its retrieval time. One challenge is theoretical justification. The addition of another degree of freedom in a model (i.e., using a second factor aside from log posterior odds) can only improve its fit to the data. But which category feature? The space of such factors is unbounded in principle, but we lack good intuitions for choosing between them. (For natural categories, plausible factors do exist, based on semantic properties and relationships.)

Another challenge is operational: how would summary information about a category accumulate? Blending is one possibility. Blending is a mechanism for aggregate retrievals, tested mainly on situations where continuous or ordered data must be handled, such as magnitude estimation. There appears to be a natural connection, however, to information about categories, such as their sizes, their compactness, and so forth, as mentioned above. Adoption of this approach would require further research, also work in progress.

Acknowledgments

The authors would like to thank Frank E. Ritter for helpful comments. We are also indebted to three knowledgeable anonymous reviewers who pointed us to relevant literature and helped us better understand the context of our research and our results.

References

Anderson, J. R. (1991). The adaptive nature of human categorization. *Psychological Review*, *98*(3), 409.

Anderson, J. R. (1996). ACT: A simple theory of complex cognition. *American Psychologist*, *51*(4), 355.

Anderson, J. R. (2009). *How can the human mind occur in the physical universe?* Oxford University Press.

Anderson, J. R., & Betz, J. (2001). A hybrid model of categorization. *Psychonomic Bulletin & Review*, *8*(4), 629–647.

Anderson, J. R., & Matessa, M. (1992). Explorations of an incremental, Bayesian algorithm for categorization. *Machine Learning*, *9*(4), 275–308.

Bothell, D. (n.d.). ACT-R 7 reference manual.

Fields, M. A., Lennon, C., Martin, M., & Lebiere, C. (2017). Priming for autonomous cognitive systems. In *Micro-and nanotechnology sensors, systems, and applications IX* (Vol. 10194, p. 1019421).

Kruschke, J. K. (2008). Models of categorization. *The Cambridge handbook of computational psychology*, 267–301.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York, NY: Cambridge University Press.

McCallum, A., & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. In *AAAI-98 workshop on learning for text categorization* (Vol. 752, pp. 41–48).

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: I. an account of basic findings. *Psychological Review*, *88*(5), 375.

Petrov, A. A., & Anderson, J. R. (2000). Anchor: A memory-based model of category rating. In *Proceedings of the annual conference of the cognitive science society* (pp. 369–374).

Pothos, E. M., & Wills, A. J. (2011). *Formal approaches in categorization*. Cambridge University Press.

Rosch, E., & Lloyd, B. B. (1978). *Cognition and categorization*. Hillsdale, NJ: LEA.

Rosch, E., & Mervis, C. B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, *7*(4), 573–605.

St. Amant, R., Goodwin, P. R., Dominguez, I., & Roberts, D. L. (2015). Toward expert typing in ACT-R. In *Proceedings of the international conference on cognitive modeling (ICCM)* (pp. 232–237). ICCM.

Thomson, R., Harrison, A. M., Trafton, J. G., & Hiatt, L. M. (2017). An account of interference in associative memory: Learning the fan effect. *Topics in Cognitive Science*, *9*(1), 69–82.

Van Maanen, L., & Van Rijn, H. (2006). An accumulator model account of semantic interference in memory retrieval. In *Proceedings of the international conference on cognitive modeling (ICCM)* (pp. 322–327).

Modeling Instruction Fetch in Procedural Learning

Bryan Stearns (stearns@umich.edu)
John Laird (laird@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

Abstract

Cognitive architecture agents execute and compile known procedures to model how humans learn procedural skill knowledge for a given task. It is often assumed that agents have fully learned how to order and condition that execution before the task begins. Is this assumption valid, or can it impair human modeling efforts? We posit that the first of the classic three phases of skills necessitates learning how to order task execution and that models should account for this process. We evaluate the effects of modeling this process using a general fetch and execute learning agent in Soar and apply it to modeling two different human studies. Our agent introduces a procedural chunking method for autonomously learning declarative memory structures by which spreading activation can guide task-specific execution order. We demonstrate that modeling the process of learning how to order task execution can significantly improve human model results.

Keywords: skill acquisition; control; phases; spreading activation; cognitive architecture; Soar; primitive elements; PROP.

Introduction

Procedural skill knowledge is considered a fundamental component of human cognition. How the human mind learns and organizes such knowledge has been of scientific interest for many years. Cognitive architectures, such as Soar and ACT-R, provide a means to improve our understanding of the mind through computational simulation of the processes thought to underly cognition. Some consensus has arisen in the community regarding certain qualities of cognition that these systems both reflect (Laird, Lebiere, & Rosenbloom, 2017).

One theory of skill learning that has been operationalized in ACT-R is the classic three phases proposed by Fitts and Posner (1967). In the *cognitive phase*, the mind learns how to understand task execution by deliberately reasoning over each cue, action, and desired outcome. Task responses might be incorrect while the correct ordering of actions is learned. The *associative phase* follows once practical task understanding is achieved. Responses to task cues become increasingly fluid with practice. In the final *autonomous phase*, those responses are automatic and occur by reflex, subject to little cognitive control or interference, such that unrelated reasoning can occur in parallel.

In ACT-R, *cognitive phase* learning is modeled via procedure compilation (Tenison & Anderson, 2016). Procedures are represented as rules, and during practice new rules are created by combining or specializing existing rules for more efficient task operation. It models *associative phase* learning by strengthening declarative memories, such that they are more readily available after practiced access. An ACT-R agent reaches the *autonomous phase* when it has learned task-specific rules that incorporate declarative knowledge, such that declarative retrievals are not needed.

Primitive elements theory (Taatgen, 2013) describes how architecturally primitive procedural knowledge can be compiled into useful, transferable skills through practice. This learning can be applied to any task using a general *fetch and execute* cycle. Knowledge of how to practice a skill is fetched from long-term declarative memory, and the indicated skill is then practiced and learned. When primitive elements theory was implemented in the Actransfer architecture, a variant of ACT-R, it produced good models of both human learning and transfer in many specific domains (Taatgen, 2013).

Primitive elements theory has also been applied to the Soar cognitive architecture, leading to the Soar PRimitive OPerator (PROP) model of primitive learning and transfer (Stearns, Assanie, & Laird, 2017). The PROP model includes an additional process of generating working memory addresses as a primitive skill necessary for instantiating general procedural knowledge. This inclusion improved power-law learning behavior to better match human data.

However, all these methods assume that *when* to fetch skills is learned before model behavior begins. This assumption is common in cognitive modeling, and underlies designs that hard-code task conditions into initial procedural knowledge. When to fetch and how to execute are learned separately, and fetching is abstracted away from models.

In this paper we consider whether that assumption neglects an important aspect of *cognitive phase* learning: learning how to arrange task execution correctly. How might models that learn skill fetching alongside execution differ from those that ignore fetching? Does this assumption impair human modeling? To shed light on these questions, we extend the original PROP model to learn to fetch *during* skill execution. We show that modeling how skills are fetched during task execution can significantly improve simulations of human behavior. For simplicity, we will refer to the original PROP model as PROP_1, and our model that learns to fetch as PROP_2.

Fetch and Execute

PROP_2 is defined for Soar, in which procedural knowledge is encoded as *if-then* rules, and describes how a task-specific skill can be converted from declarative into procedural knowledge through repeated practice of a task-general *fetch and execute* cycle.

PROP_2 progresses through the three phases of skill learning. In the *cognitive phase*, the agent knows rules for the process of fetching and executing instructions, but does not know rules for when specific instructions should be fetched and applied. Until it learns to fetch, a process we describe later, fetching is random and can retrieve non-applicable in-

structions. After the agent gains enough experience to always fetch applicable instructions, it is in the *associative phase*. In this phase, it learns increasingly task-specific rules that reduce instruction execution latency. Eventually the agent learns rules to perform a task automatically without fetching, and is then in the *autonomous phase*.

We assume that instructions are stored in long-term declarative memory by some prior learning process. Interactive task learning (Laird, Gluck, et al., 2017) is one means by which an instructor can teach instructions to an agent.

Instructions include three components: conditions, actions, and any literal values used within them. Conditions and actions describe primitive assembly-like operations supported by the architecture, and these are uniquely identified by the working memory elements they use. For example, instructions might load the string "hello" (a literal value) into memory element A, test equality of values in elements A and B (a condition), and then copy the value from B to C (an action).

After fetching, the agent executes an instruction by first evaluating the instruction conditions. Initially this is done with innate primitive rules. One rule fires to test the first condition, another tests the second condition, and so on. If all are true, additional rules execute the actions. If any are false, the agent returns to fetching to attempt different instructions.

When the agent executes a sequence of instructed operations, it automatically learns rules that combine those operations together, as pictured in Figure 1. This process drives the bulk of learning during the *associative phase*.[1] Sequentially associated rules are combined into new rules, which fire to execute instructions in fewer steps than required by primitive rules. These rules are general and can transfer to execute any instructions that invoke the same architectural operations, but their structure is specific to experienced task operations.

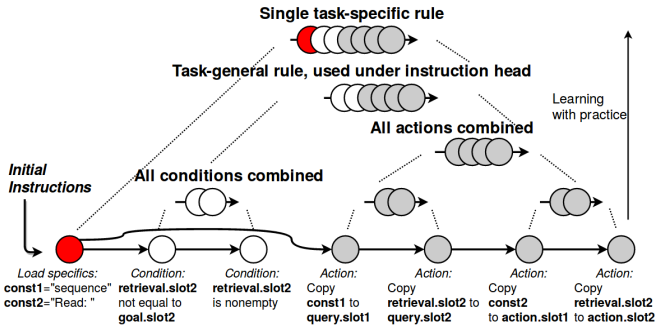


Figure 1: Hierarchical clustering of primitive memory operations, adapted from (Taatgen, 2013). The architecture iteratively combines seven primitive task-general operations through practice into a single task-specific rule. In this example, actions query for the next item in a sequence, while printing the current value.

Taatgen (2013) showed that hierarchical learning of primitive memory operations provides a good model for human skill learning and transfer in many domains. For good transfer, it is important that rules are built *gradually* over iterations of execution. A new rule only takes effect after multiple learning attempts so that only rule combinations that appear often across instructions are used within the skill hierarchy.

[1] In agreement with ACT-R, our model also considers declarative strengthening to be *associative phase* learning.

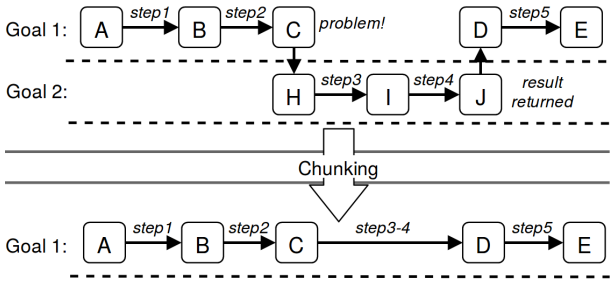


Figure 2: Soar chunking. Goal 2 is created as a subgoal of Goal 1. Goal 2 decision making eventually derives a result that allows Goal 1 to continue. Chunking learns this result as a rule that recreates the solution (step3-4) the next time the scenario arises.

Associative rule composition eventually builds a single task-specific rule equivalent to the complete fetched instruction set. Once learned, this rule will execute the instructed conditions and actions by reflex without the need for reasoning over fetched instructions. An agent that primarily performs a task by using such rules is in the *autonomous phase*.

This process defines how task rules are learned, and ignores how the agent knows what to fetch and execute at a given moment. But, as we will show, the same methods can be used for learning what to fetch. In the next section, we describe gradual procedural learning in Soar, the mechanism we use for hierarchical associative learning and for learning to fetch.

Gradual Procedural Learning in Soar

ACT-R learns procedural knowledge by a hierarchical compilation process similar to Figure 1. In contrast, Soar learns procedural knowledge by summarizing problem solving.

Soar manages a stack of agent goals and corresponding working memory partitions. Newer subgoals in the stack represent subproblems of earlier goals. As shown in Figure 2, whenever agent reasoning connects declarative knowledge from a subgoal's memory to a higher goal's memory ("returns a result"), the architecture summarizes subgoal decision making by creating a new rule that recreates that result whenever the situation that led to the subgoal arises again. This learning is called *chunking* in Soar, and learned skills are called *chunks*.[2] Chunking is normally one-shot, and once a result is returned, the corresponding chunk will preempt future subgoal problem solving. To implement our model, we extended Soar to include gradual chunking.

To chunk gradually, the architecture tracks the number of times a specific chunk is submitted for creation and only adds that chunk to procedural memory if that number passes a parameterized threshold.[3] Setting the threshold to 1 is equivalent to standard one-shot chunking. A threshold of 2 would

[2] Not to be confused with ACT-R declarative *chunks*.

[3] Other methods can easily be used in place of creation counts, but this is sufficient for our purposes here.

require a subgoal to be solved twice, and so on.

To build the skill hierarchy shown in Figure 1, the $PROP_2$ model includes innate rules that explicitly combine pairs of invoked operations together as results for chunking after they are executed. These combined operations can then be used to apply instructions in future iterations of execution.

Learning to Fetch in Soar

An agent might have many instructions in long-term declarative memory that it could fetch at any time. When it needs to execute and learn a new skill, how does it know what to retrieve? Instructions will not be applicable unless their described conditions are true, but how can the agent know if the conditions for specific instructions are true until after it fetches and evaluates them? Without a method for biasing memory retrievals toward instructions that are likely to have matching conditions, all of declarative memory might have to be searched before usable instructions are found. While the order of operations can be hard-coded into rules, we argue that learning what to retrieve is an important aspect of *cognitive phase* learning.

The Actransfer architecture controls fetching by precalculating which conditions were true for each instruction that might be retrieved, and boosting the activation for instructions with satisfied conditions (Taatgen, 2013). The agent then fetches instructions with the highest activation value.

$PROP_1$ implemented instruction fetching by deliberately controlling the fetch sequence. Declarative knowledge of the correct instruction sequence for a given task (e.g. "$Skill1 \rightarrow Skill45 \rightarrow Skill2 \rightarrow Done$") was provided to an agent at the same time as those instructions. The agent then iteratively fetched each element of that sequence to perform a task. This method required the agent to constantly track its current position in that sequence. A problem with that approach is that the agent loses track of its position in the sequence if it enters the *autonomous phase*. The task-specific rules that drive behavior in that phase act by reflex outside fetching and the agent lacks meta-knowledge of what rules it fires. Thus, the agent will not know to update its sequence position. When it does need to fetch, it will resume where it last left off in the declarative sequence, and will sequentially fetch and unsuccessfully evaluate all operations that were just performed autonomously until it catches up to the actual state of the task. Deliberate fetching becomes a performance bottleneck by requiring cognitive control over selecting each task step, even when that step has already been performed.

Neither of these approaches explains how skill fetching is learned. Rather than assuming activation boosts as in Actransfer or relying on a fixed fetching sequence as in $PROP_1$, we create a more comprehensive model that learns how to fetch instructions. We implement this by using Soar chunking to learn connections for spreading activation.

Primitive elements theory proposed using spreading activation to guide instruction fetching, though this was not implemented in Actransfer (Taatgen, 2013). By this method, when a query to long-term declarative memory has multiple possible results, the memory with the highest activation is retrieved. Spreading activation increases the activations of memories associated with the current working memory context. Working memory elements connected to long-term memory elements boost the activations of those memories according to connection weights. If those long-term memories are connected to other long-term memories, that boost spreads, with some decay, to also increase their activations, and so on to other memories.

Inspired by primitive elements theory, and the recent addition of spreading activation in Soar (Jones, Wandzel, & Laird, 2016), we developed a novel model for learning to fetch. This model follows three constraints of Soar theory: First, only working memories that also exist in long-term memory can be sources of spreading activation. Second, Soar models a fan-effect by normalizing a memory's spread over the number of its descendants in the long-term memory graph. Third, such working memories are either created through long-term memory retrievals or through chunks.[4]

To learn spreading, $PROP_2$ learns to create or remove long-term memory elements within working memory, causing spread according to the first constraint. Each spread source corresponds to an individual condition described by an instruction, and spreads to that instruction. The agent learns to create a spread source whenever the corresponding condition is true and remove it when it is false. Thus, instructions with the most true conditions are favored for retrieval.

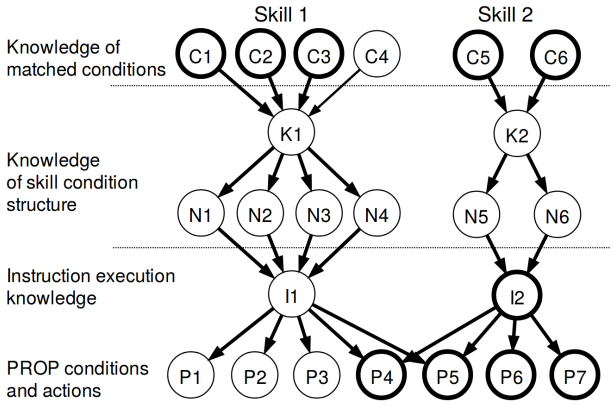


Figure 3: Declarative structures for skill fetching in Soar. Nodes in bold are present in working memory. Spread from these is normalized over the number of a rule's conditions. Since condition C4 is false, I1 receives 3/4 spread while I2 receives 2/2. I2 is then fetched.

But the agent should ideally retrieve instructions with *fully* satisfied conditions. If all else is equal, instructions with only six of seven conditions met should receive less total activation than those with two of two conditions met. Therefore, according to the second constraint, we structure long-term memory so that spread normalizes over the number of conditions in a set of instructions, as shown in Figure 3. In the figure, three

[4]Creating such memories through chunks is a feature of Soar 9.6, developed by Mazin Assanie.

of four conditions are known to be satisfied for Skill1, and two of two conditions for Skill2. The execution knowledge for Skill1 thus receives 3/4 normalized spread while that for Skill2 receives 2/2 spread. Skill2 is therefore fetched.

To satisfy the third constraint, we use chunking to learn *when* to create knowledge of satisfied conditions. When evaluating the conditions from fetched instructions, if the agent finds a true condition, it returns declarative knowledge of that result. After instructions have been fetched and evaluated enough times over the course of a task to satisfy the chunking threshold, Soar automatically chunks a rule that recreates this result whenever the condition is met. As soon as that condition is no longer met, the chunk no longer matches, and the spread source is removed from working memory.

For simplicity, we provide long-term declarative knowledge of conditions at the same time as instructions, structured in the format shown in Figure 3. Our evaluation is concerned with learning to create knowledge of matched conditions within working memory.

Our agent begins each fetch with an open query for instructions that can be biased by spreading. Results will be random at first before conditions are learned. If our agent finds that it retrieved non-applicable instructions, it reverts to cognitive control for fetching by retrieving and following the explicit declarative fetch sequence. As the agent learns conditions through experience, spread provides sufficient bias such that controlled fetching becomes unnecessary.

Parameters for Skill Fetching

We have described our task-general design for learning skill fetching and execution. Our aim is to evaluate the importance of modeling the fetch process. We define two boolean parameters for modeling fetching, which in the $PROP_2$ model determine what kinds of rules an agent learns.

The first parameter is either **LEARNED** or **KNOWN**, and determines whether an agent learns fetch order during task execution. If fetching is LEARNED, the $PROP_2$ agent will learn rules for spreading during execution. If fetching is KNOWN, the agent will *always* fetch through cognitive control by following a declaratively known fetch sequence.

The second parameter is either **AUTO** or **DELIBERATE**. If set to AUTO, the agent eventually learns a single rule that will perform task operations by reflex (the top-most composition depicted in Figure 1), bypassing declarative instructions and cognitive control. If set to DELIBERATE, this final composition step is prevented, so that execution must *always* be deliberate and fetching cannot be bypassed.

Evaluation

We evaluate these parameters against human behavior using two human domains: a text-editors task Singley and Anderson (1985) and a mental arithmetic task (Elio, 1986). The editors task examines transfer, while the arithmetic task focuses on the learning curve. These have already been modeled in Actransfer, allowing us to compare our $PROP_2$ model results with those from the alternate fetch/execute paradigm. Actransfer is a KNOWN-AUTO model, since it assumes fetching order (by hard-coding activation behavior) and allows task rules to be fully learned, so ideally our model should be similar to Actransfer when using KNOWN-AUTO.

We implement the same Actransfer agent designs, using supplementary materials from (Taatgen, 2013), including the same declarative instructions and the same simulated timing for memory retrievals and vision/motor latencies. We also model time using the same 50 msec per decision as is common in Soar and ACT-R.[5]

To select the gradual chunking threshold, we performed a threshold sweep (not shown) to find the value that provides the closest fit to the Actransfer model for comparison. Matching differences in Actransfer model learning rates, we use a threshold of 48 for editors task models, and 10 for arithmetic task models. We also match Actransfer by averaging performance over 12 trials for the editors task and 8 trials for the arithmetic task. These models are fairly deterministic, and variance is low.

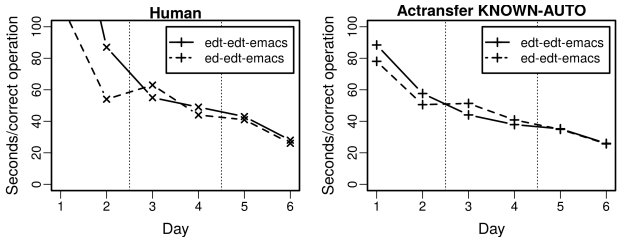
We compare our results with average human performance rather than that of individuals. Taking the average abstracts away any individual differences among humans that would arise from lifelong learning experiences that preceded participation in the studies. Modeling average human performance is our first step, and modeling individual differences is beyond the scope of the current study, although we plan to study it in the future.

Editors Task

In the editors task, typists modified documents according to written edit directions. Example directions are to replace one word with another or to delete a sentence. Three keyboard-only editors were used with which participants had no prior experience: ED, EDT, and EMACS. These use different keyboard commands, and ED and EDT also differ from EMACS by being simpler single-line editors. The experiment took place over six days, with some participants switching editors after two days to test transfer. If a participant spent two days each on ED, EDT, and EMACS in that order, we call this case ED-EDT-EMACS. If initial performance in EDT was faster after using ED than when using EDT on day one, this indicated transfer to EDT. We focus on transfer of ED to EDT-EMACS, but other editor permutations show similar results.

Figure 4a shows results from the human experiment. Performance on EDT after two days of ED is almost as good as after two days of EDT, indicating substantial transfer. There is similarly significant transfer to EMACS on day five. (EMACS users required about 80 sec on day 1, not shown). Figure 4b shows the Actransfer model. Model performance is fast during days 1-2, but transfer trends are the same. Figure 5 shows the parameterized $PROP_2$ models. Transfer is similar among all models in Figures 4 and 5.

[5]Soar uses decisions to carry out retrievals, and we replace the 50 msec from those decisions with retrieval time.



(a) Human performance (b) Actransfer model

Figure 4: Human data from (Singley & Anderson, 1985) and the Actransfer model from (Taatgen, 2013), demonstrating transfer between editors.

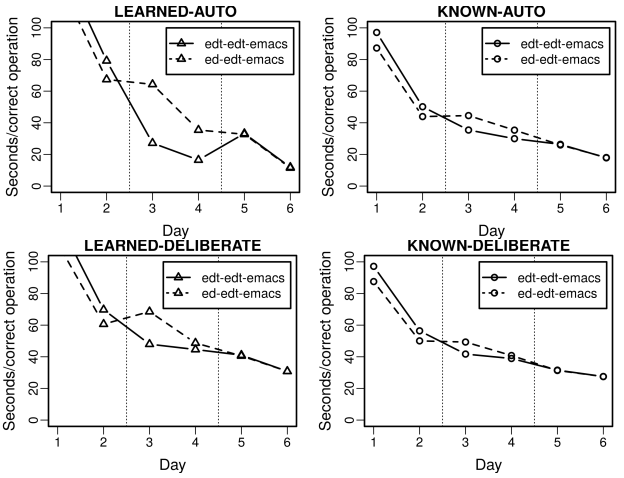


Figure 5: $PROP_2$ models of editors, varying over LEARNED (left column), KNOWN (right column), AUTO (top row), and DELIBERATE (bottom row).

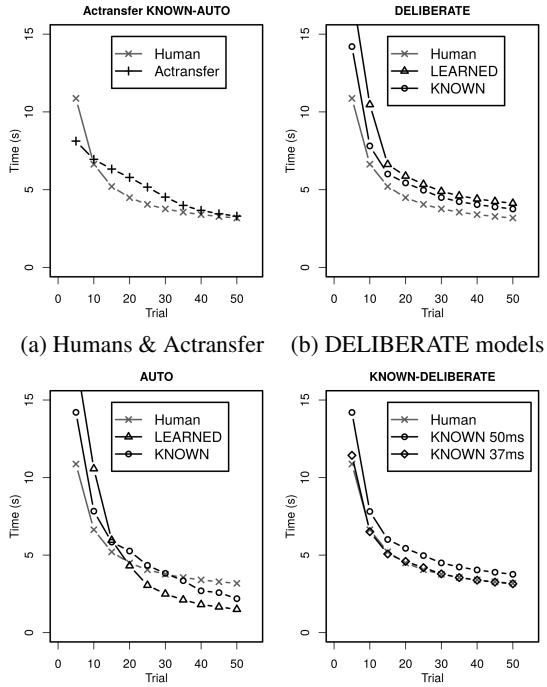
In Figure 5, observe that there is a significant difference between LEARNED (left column) and KNOWN (right column) results, primarily in slower LEARNED performance on days 1-3. These are cases when non-controlled fetching retrieves mostly random instructions. Interestingly, after fetch learning, LEARNED models are not far behind the KNOWN models in performance, demonstrating that LEARNED models learn fetching and execution simultaneously.

Also notice that AUTO models (top row) achieve super-human performance by day 6, at under 20 sec. Human and Actransfer models, by contrast, end near 30 sec, as do our DELIBERATE models (bottom row). The Actransfer KNOWN-AUTO model uses precalculated activation to fetch, and does not exhibit this phenomenon. But we surmise that Actransfer is slower than our AUTO on day 6 due to activation noise, a part of that model that can also cause incorrect fetching.

Overall, we see that LEARNED-DELIBERATE (bottom left) most closely matches humans by accounting for slower performance during the first days (due to LEARNED), while finishing at the correct performance on the last day (due to DELIBERATE). Performance on days 1-3 is too fast for EDT but slightly slow for ED, implying the instructions for EDT should be more complex and those for ED slightly less so.

Arithmetic Task

In the arithmetic task, human subjects memorized a mental arithmetic algorithm and applied it to provided inputs for 50 trials. For brevity we omit transfer details and focus on the learning curve.



(a) Humans & Actransfer (b) DELIBERATE models

(c) AUTO models (d) Best human model

Figure 6: Models of the (Elio, 1986) arithmetic experiment.

Figure 6a shows results for humans and the Actransfer model. Only the power-law fit to human performance was available from the original study, and this is what is shown as the human model.

Figure 6b shows our LEARNED- and KNOWN-DELIBERATE models. We first note the power-law performance of $PROP_2$, which is from having to learn how to access working memory for tasks, inherited from $PROP_1$ (Stearns et al., 2017). We also see clearly how the LEARNED model catches up to the KNOWN model after 15 trials. It does not catch up entirely, due to the fuzzy nature of using activation.

Figure 6c shows our LEARNED-AUTO and KNOWN-AUTO models. As with the editors task, the AUTO parameter results in eventual super-human performance. The KNOWN-AUTO learning curve is not smooth, due to times when autonomous task-specific rules invalidate the controlled fetch sequence, which must then be stepped through to catch up to the task state. But we observe that after trial 20 the LEARNED-AUTO agent acquires enough fetch experience to not require further deliberate fetch control.

We also observed that while model time incorporates many factors, such as retrievals and motor latency, the amount by

which KNOWN-DELIBERATE behavior differs from human performance is a multiple of decision cycle time. Setting decision cycle time to 37 msec results in almost an exact fit to the human curve, as shown in Figure 6d.[6] While not a standard timing, this fits neural modeling that predicts cycle times of approximately 40 msec for simple actions (Stewart, Choo, & Eliasmith, 2010). Table 1 shows the mean-squared-errors of various timing models compared with the human model.

	Actransfer	**50 msec**	**40 msec**	**37 msec**
MSE	1.270	1.695	0.177	**0.0382**

Table 1: Mean Squared Errors for Actransfer and KNOWN-DELIBERATE models with varying simulated times for decisions.

Discussion

The above parameters reflect Fitts and Posner's (1967) three stages of skill learning in our model. The *cognitive phase* is represented when fetch LEARNING is enabled. Hierarchical compilation of instruction execution demonstrates the *associative phase*, and allowing AUTO task-rules to be compiled leads to the *autonomous phase*.

We model *cognitive phase* learning with a novel use of Soar chunking by which chunks create or retract sources of spreading activation. This general learning method could be applied whenever there is an occasion for learning context-appropriate retrievals, not just fetch modeling.

In the editors task, LEARNED fetching accounts for initial human behavior in ways KNOWN fetching could not. However, in the arithmetic task, KNOWN models were most accurate. We believe this reflects the natures of the tasks. Human subjects performed the arithmetic task *after* being trained in the algorithms, while editors subjects memorized only an editor's individual keyboard commands prior to experimentation. Subjects performing the arithmetic experiment should therefore have already mostly completed the *cognitive phase*, which is not the case for subjects in the editors experiment.

Though AUTO allows faster execution, we notice that DELIBERATE achieves the closest human performance in both experiments. Within our model this implies that humans do not perform these tasks entirely by reflex after training, but continue to reason over each step. This might imply that the *autonomous phase* is more appropriate for modeling motor skills, as Fitts and Posner were evaluating, rather than dominantly cognitive skills.

Our arithmetic task model is almost identical to the human model when decision cycle time is just under 40 msec. By contrast, changing cycle times has little effect on the editors model, since it performs at the scale of 100 sec, largely due to memory retrieval times. Editors agents also employ a higher chunking threshold than used by arithmetic agents, implying that human processing is more complex for the editors task than for the arithmetic task compared to our models, which also makes sense given the different time scales. The appropriate complexity of task models and the validity of 40 msec cycles for primitive skills are beyond the scope of this paper. However, they present intriguing avenues of study.

We therefore conclude that consideration for the stages of learning and these fetching parameters is critical for human modeling. Whether an agent learns fetching or assumes it to be already learned should reflect the task being modeled. The fetch process of choosing what to execute is one by which the human mind controls its own behavior. Learning that control can be important for even simple modeling, as we have demonstrated, but understanding the theory of that learning could play an important role in unraveling the mysteries of human cognition.

[6]This would not be achieved by changing the chunking threshold, as that alters the learning curve shape in addition to scale.

Acknowledgments

The work described here was supported in part by the Office of Naval Research under Grant Number N00014-18-1-2010. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the ONR or the U.S. Government.

References

Elio, R. (1986). Representation of similar well-learned cognitive procedures. *Cognitive Science*, *10*(1), 41 - 73.

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

Jones, S. J. M., Wandzel, A. R., & Laird, J. E. (2016). Efficient computation of spreading activation using lazy evaluation. In *International conference on cognitive modeling.*

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.

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.

Singley, M. K., & Anderson, J. R. (1985). The transfer of text-editing skill. *International Journal of Man-Machine Studies*, *22*(4), 403 - 423.

Stearns, B., Assanie, M., & Laird, J. E. (2017). Applying primitive elements theory for procedural transfer in soar. In *International conference on cognitive modeling.*

Stewart, T. C., Choo, X., & Eliasmith, C. (2010). Dynamic behaviour of a spiking model of action selection in the basal ganglia. In *International conference on cognitive modeling* (pp. 235–40).

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471.

Tenison, C., & Anderson, J. R. (2016). Modeling the distinct phases of skill acquisition. *Journal of experimental psychology. Learning, memory, and cognition*, *42 5*, 749-67.

Modeling Visual Search in Interactive Graphic Interfaces: Adding Visual Pattern Matching Algorithms to ACT-R

Farnaz Tehranchi (farnaz.tehranchi@psu.edu)
Department of Computer Science and Engineering
Penn State, University Park, PA 16802 USA

Frank E. Ritter (frank.ritter@psu.edu)
College of Information Sciences and Technology
Penn State, University Park, PA 16802 USA

Abstract

We provide an update on JSegMan, an interactive system to extend the ACT-R cognitive architecture to interact with dynamic interfaces based on the screen contents and generating input for the operating system directly. Current ACT-R models typically interact with the world through ACT-R's device interface—an abstract representation of the world that is based on a simulated Lisp environment provided with ACT-R, or by instrumenting interfaces. In JSegMan, computer vision pattern matching algorithms and visual patterns extend the ACT-R cognitive architecture. With JSegMan, models directly move the cursor on the screen, click on application GUI objects on PCs, and type through the use of existing Java libraries. Implementing users' visual search strategies and input abilities for different visual objects enables the detailed modeling of interactive tasks on any interface. The visual pattern matching algorithms serve two goals: to simulate user behavior in interactive tasks and to create representations of visual stimuli. We tested our visual pattern matching approach by using it with an existing model for a long spreadsheet task. We found that the revised model more accurately predicted a 20-min task by entirely performing the task on an uninstrumented and unmodified interface.

Keywords: Cognitive model; Cognitive architecture; Human-computer interface, interaction; Perception and motor output; Computer vision; Simulated eyes and hands.

Introduction

Cognitive architectures are programming languages specifically designed for modeling unified theory of all human cognition, such as Soar (Laird, 2012; Newell, 1990) and ACT-R (Anderson, 2007). Using models as users have been envisioned before (e.g., Byrne, 1994; Lohse, 1997), but has not yet been widely applied. These models require simulating an interactive behavior.

ACT-R's interactive behaviors include moving the cursor, clicking, and typing. Although cognitive architectures such as ACT-R provide powerful and flexible frameworks for modeling general human behavior, only a portion of their capabilities are often used to capture many complex real-world behaviors (Fu et al., 2003), because the skills can only be applied with unmodified ACT-R in specialized windows provided with ACT-R. Additionally, cognitive models need to interact with interfaces to be useful in human-computer interaction (HCI) (Byrne & Kirlik, 2005; Kieras, 2009; Ritter et al., 2000).

A Cognitive Model Interface Management System (CMIMS), which is an extension of the User Interface Management System (UIMS) concept (Ritter et al., 2001) is an approach to provide models access to interfaces so that cognitive models can be applied more routinely in HCI.

ACT-R/PM (Byrne & Anderson, 1998) is a CMIMS that allows ACT-R models to interact with interfaces built within a special Common Lisp window. Another successful version of the interactive model is Salvucci's (2006) driver model. Salvucci also introduced an ACT-R system with an implementation of the ACT-R cognitive architecture in the Java programming language. It can be used as a library in other ACT-R projects (Salvucci, 2009, 2013).

In all these approaches, it will be difficult to reuse these models (their task knowledge) because of the nature of the embedded task environment—the models can only interact with their instrumented interfaces. The interfaces that have been modified to provide the models access to information on the display in that application. Also, the interfaces have been adjusted to accept commands directly from the models. Even in JSON ACT-R, which makes a general approach available for these modifications (Hope, Schoelles, & Gray, 2014), reconfiguring the connection is challenging.

We take inspiration from SegMan, segmenting the screen to different features (e.g., based on color) and manipulating interface usage. SegMan was productive and was used to interact with a wide range of interfaces (Ritter, Kukreja, & St. Amant, 2007; St. Amant et al., 2005; St. Amant & Riedl, 2001). However, SegMan is hard to use and maintain at this point. We have also explored providing a connection to any task in Emacs, ESegMan, such as pressing a key, moving a cursor, clicking a mouse, and moving attention (Tehranchi & Ritter, 2017). ESegMan could only manipulate Emacs window and is limited to Emacs.

These results have led us to design JSegMan, a system written in Java that extends ACT-R to provide models with interaction with any interface on a PC, and thus the world.

We report elsewhere its hands model that provides typing and mouse interaction (Tehranchi & Ritter, 2018). JSegMan creates a way to interact with all interfaces using an extended Java library to input motor commands (keystrokes, mouse moves, and mouse clicks).

We report here the first vision system for cognitive models to see objects on the screen using pattern recognition and other computer vision algorithms.

As an example application, we use ACT-R and JSegMan to perform the Dismal spreadsheet task using an existing large (500 rule) ACT-R model (Paik, Kim, Ritter, & Reitter, 2015). The revised model with JSegMan predicted the response time more accurately while, importantly, using the same but unmodified interface that the human subjects used. Finally, we provide conclusions, lessons, and insights.

Visual Attention and Environment

There are theoretical and empirical challenges in understanding visual attention and implementing its mechanism. Constructing a system to simulate attention based on eye movement data such as fixation and saccade can enhance our knowledge of perceptual-motor aspects of the human system. Besides cognition, perception, and action are also essential for resembling the models' behavior closer to the human behavior. The focus of cognitive analysis is often on cognition and response time, rather than an analysis of interaction and knowledge related to the interaction. Given the visual nature of most current user interfaces, the vision module is relatively central in modeling most HCI tasks. The vision module is used to determine what ACT-R/PM "sees" (Byrne, 2001).

Each object on display will be represented by one or more features in the vision module's icon. The vision module creates chunks from these features that provide declarative memory representations of the visual scene. These chunks are visual location and visual object types. Production rules' constraints can match chunks. After the vision module creates a chunk representing an object, visual attention must be directed to the location of that object, through a visual location chunk.

Anderson et al. (1998) introduced the visual hunt-feature and found-target in which there are the equivalents of move attention and move the cursor in current ACT-R. The vision module move-attention operator shifts attention to a location; ACT-R/PM must have representation for those visual locations.

Salvucci (2001) distinguished eye movements and shifts of attention. The ACT-R visual mechanism mainly considers moving attention to be the same as a saccade and reflects a top-down search without backtracking. Furthermore, the visual mechanism skips items that do not have the same features as the target features. Kieras and Meyer (1997) in EPIC predicted that eye movement patterns at least in their task, menu search, are 50% top to bottom and 50% random (Hornof & Kieras, 1997). The EPIC inspired visual mechanism looks for the target object under the current location rather than starts at the top of the display; current ACT-R does not implement this mechanism.

Static Environment: In this kind of environment, the interface displays the application semantics with which the user interacts directly and immediately. No other processes can modify the environment and there will be only one level of commands. For example, we have gathered data by showing users pictures in a PDF document and having them click on a target object. The object does not change when clicked upon.

Interactive Environment: In this kind of environment, the interface content only changes when the user provides input. For instance, in Emacs, the display typically only changes based upon the user input.

Dynamic Environment: These user interfaces often have a high degree of interaction and are complex, and may be non-deterministic to the user. The boundary between application and user interface is difficult or impossible to control, and what the user can see may change without the user's input. The screen content changes based on both user input and time. Therefore, it becomes problematic to decide if this interaction should be handled by the model or in the application level (Soegaard, 2013).

The JSegMan system is intended to allow modelers to participate more directly in all of these environments, particularly the interactive and dynamic environments to study computer behavior.

In this paper, JSegMan has been examined in the Dismal spreadsheet mode of Emacs, which requires a nested level of commands (Ritter & Wood, 2005). Dismal is a spreadsheet developed for Emacs that was designed to gather and analyze behavioral data. Users can interact with the spreadsheet similar to other Emacs windows and also with Dismal function calls. Dismal is an interactive environment that its interface components change by user input and commands.

Visual patterns in JSegMan are small images, a representation of an object in the visual scene. Figure 1 shows different patterns on the screen. Visual patterns in JSegMan are equivalent to visual location and visual object chunks in ACT-R. ACT-R can access all required objects of the current visual scene by defining the patterns as independent patterns or a logical mix of patterns. JSegMan needs to use multi-part patterns to interact with more complicated environments. Modelers create required visual patterns for JSegMan.

In the Dismal model, to find a spreadsheet cell, as shown in Figure 1, the user first finds the column and then looks for the row. To implement this action in JSegMan, we have to find the pattern, re-set the overlap region, and find the new pattern (e.g., the row) inside a pattern. This nested pattern finding will make the models more human-like by requiring additional knowledge and steps that take additional time and allow the opportunity for additional errors. To add this functionality, the matching algorithm needs some higher level of built-in logic.

In JSegMan, the target visual pattern is compared with all the same size available frames in the screen. It moves the comparison frame one pixel at a time, a constant distance in each saccade, from left to right and top to bottom.

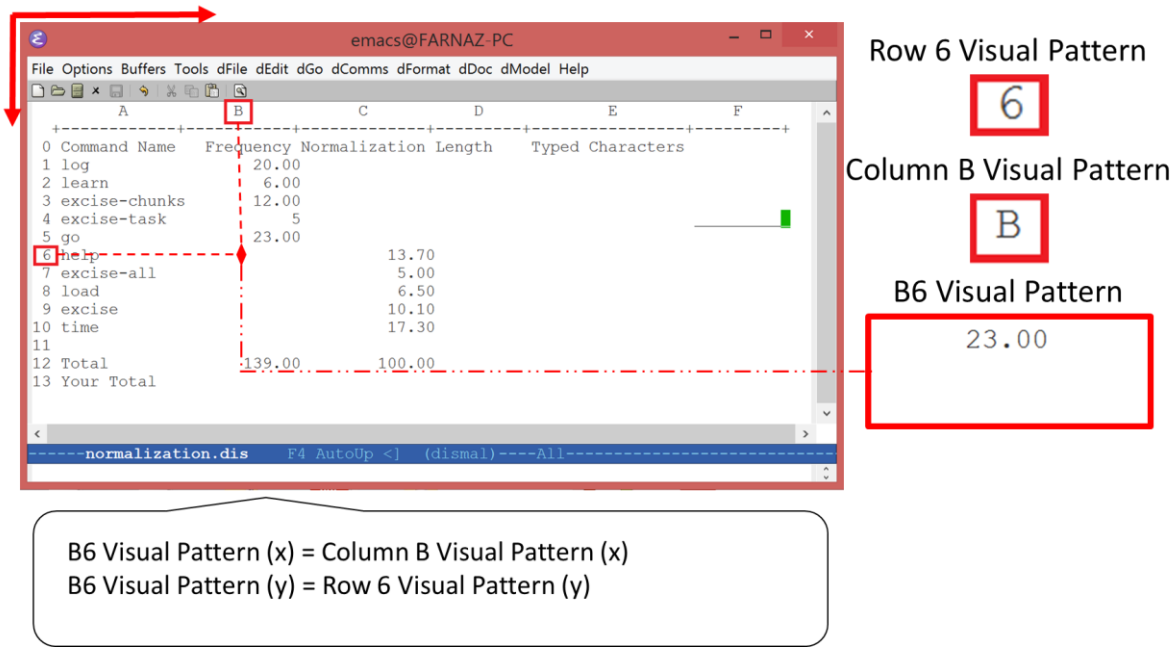


Figure 1. Defining visual patterns in an interactive environment.

This approach uses patterns instead of regular visual objects. Therefore, visual object features in ACT-R such as size and location are not practical for the model. Consequently, any changes in the visual scene cannot affect patterns such as changes in screen resolution and size. The model can still find the target visual object because visual attention does not change by screen-x and screen-y. Instead, the model finds/re-finds the pattern in the visual scene. From Java process, JSegMan can use these features and update the visual representation in ACT-R. This approach is independent of the task environment's system because the location of task environment on the screen does not change the underlying patterns.

To support working with interaction interfaces, we propose two methods: (a) The model always has a knowledge of the dynamic environment because the model has to respond appropriately to manipulation of said environment (Anderson, Farrell, & Sauers, 1984; Soloway & Johnson, 1984), and (b) The model uses pattern matching logic. The next section explains these methods in details.

Visual Pattern Matching Algorithms

We used two Java packages: (a) Robot and (b) Sikuli (Yeh, Chang, & Miller, 2009) to develop the JSegMan system. JSegMan functions are divided into two primary sections. The first, hand simulation, includes motor module methods, *move cursor to*, *get mouse coordinates*, *handle click*, *handle keypress*, and *type word*. The second section, the eye simulation, includes *move attention*, *process display*, and *print visicon* (a debugging command).

Java Robot is used to automate operating system actions such as clicks and typing. The Robot package implements actions from the ACT-R motor module, such as moving the mouse as moving the attention pointer by ACT-R, clicking a mouse, and pressing keystrokes.

The experimental environment is an application GUI that contains the information of visible objects such as labels, text fields, images, buttons, links, radio buttons, and toggle buttons. With Sikuli, we have access to all these objects through the screen bitmap and can define them as Java objects (Kasper, Correll, & Yeh, 2014; Yeh, Chang, & Miller, 2009).

With Sikuli, cognitive models can identify and control GUI components, anything the end users could interact with and see on the screen (not just from Java applications), and also could support reading from text recognition (OCR). More specifically, it uses GUI screenshots for searching patterns to direct mouse and keyboard events in contrast with seeking screen locations, which may change during the experiment. Also, it utilizes Template Matching, a pattern-matching algorithm in OpenCV, an open-source computer vision library in which a pattern (small image) is compared against the overlapped image regions (the computer screen). Algorithm 1 summarizes the methods. Both the display and the visual pattern pixels matrix are the input of matchTemplate (Bradski, 2004). In the end, it returns the location with a higher matching value. This algorithm has been implemented in Sikuli.

Algorithm 1: Visual Pattern Matching	
1	Input: Visual Pattern pixels matrix Display pixels matrix Output: Result matrix in the size of the display, maximum value of the Result matrix
2	matchTemplate(Display, Visual Pattern, Result, CV_TM_CCOEFF_NORMED);
3	Return MaxLocation(Result);

First, JSegMan loads the visual patterns by processing the display. Then, the model uses pattern-matching algorithms

to find the pattern on the screen and then move the visual attention. The current visual scene is a screenshot of a computer screen.

Patterns should be distinct enough for JSegMan to be identified uniquely. The pattern-matching algorithm is color sensitive. There are some suggested approaches to use grayscale (e.g., Kuchhal, 2014). When JSegMan is looking for a visual pattern on display, one or more locations may be matched by the pattern matching process, depending on the number of objects on the display, display complexity, and the production rules' constraints.

The JSegMan process will check the pattern existence. When the JSegMan process is running, the ACT-R process pauses and resumes when JSegMan finishes the eye and hand simulation. Therefore, the JSegMan task completion times will not affect the ACT-R theoretical response time, but the system together does run modestly slower in real time. Currently, JSegMan takes 195.5 ms for pressing a keystroke (Burns, Ritter, & Zhang, 2016) and the eyes simulation delay takes 500 ms to do search tasks; JSegMan searches up to 1000 ms to find patterns in real time. If the JSegMan process cannot find the pattern, it will return an error message and pause the ACT-R process.

Furthermore, ACT-R passes a Value slot of the vision object, the name of the pattern, to the Java process. Finally, the task environment responds successfully to each of the requests made by the ACT-R model, and the model is able to create and attend to objects within the dynamically/interactively changing visual scene.

Figure 1 distinguishes logic pattern matching and regular matching. For example, to shift attention to cell B6 in the spreadsheet, we can either define a unique independent pattern for B6, the *B6 visual pattern* or define two patterns: one for its column and one for the row. Then, JSegMan calculates the exact location of B6 and move there directly. By collecting not only the reaction time data but also the eye-tracking data, we can predict how humans find a cell in the spreadsheet environment, and image recognition will have a more natural approach. In the Dismal model, the cell's B7 visual pattern is dependent on the B6's cell value. The Sikuli screenshot search engine can match the B7 visual pattern after the change in B6 is affected because the B7 visual pattern contains the B6's cell value. Using a logic pattern matching methods eliminates this dependency.

The Sikuli script operates only in the visible screen space and does not work on invisible GUI elements, such as those hidden beneath other windows, in another tab, or scrolled out of view. For instance, moving the cursor can cause a pop-up description which will block some patterns. Additionally, any UI-specific interaction, such as clicking the sidebar to scroll down the page, can be implemented by Sikuli classes. Further details, installation documents and instructions, and example models can be found on the project's website[1].

Revising the Dismal Model

To demonstrate the application of JSegMan, we tested our system with the Dismal model (Paik et al., 2015). In this study, data from 30 participants were collected while completing 14 subtasks of the Dismal spreadsheet task for mouse users. The Dismal spreadsheet subtasks were related to declarative and prosedueralize knowledge. The models' performance and participants' performance comparison was not completely realistic because the detailed trace of hand, fingers, and mouse movements were not modeled in much detail.

In our first attempt using JSegMan hands simulation, several missing keystrokes in the Paik et al.'s (2015) Dismal model had to be added, and the hands and fingers position on the virtual keyboard had to be adjusted. We were able to implement all the motor actions related to the keyboard in the original model. By redefining some of the keystrokes, we made more realistic key press actions in the virtual keyboard in ACT-R. For some keys that were not reachable by either the left or right hands, there had to be a request to the motor module to adjust the hand position. We added these requests to the original model.

In our analysis of the output of the model the keystrokes and mouse moves, we found missing actions such as clicking, including of which increased the total task time when compared to previous reports.

The Dismal spreadsheet environment is an interactive environment that changes based on user input. For instance, the *B6 visual pattern* in Figure 1 contains B5 cell's value. JSegMan simulates Dismal model eyes and hands correctly with this modification. The structure of the original Dismal model is based on pre-knowledge. Therefore, all individual patterns should be defined in advance without using the logic pattern matching strategy. The declarative chunks have been defined to move attention directly to cells rather than rows and columns. All patterns of cells are similar to the *B6 visual pattern* in Figure 1. We adjusted 162 declarative chunks in the original Dismal Model by adding a new slot for visual objects. In addition, to model eye movements, we added 52 new visual objects and visual locations. When JSegMan run with the Dismal model, we noticed a few missing sub-tasks (because the model did not produce a complete solution in the spreadsheet), so three new declarative chunks were added to the original model. The sequence of the subtasks for the visual module is not correctly interpreted. Therefore, we added a new constraint for the production rule that requests a new visual location be placed in the visual-location buffer.

Table 1 shows how the response time has been affected by our modification while the model learns over four trials. Besides proving the functionality of the JSegMan Dismal model, we were able to fit the Dismal model slightly better to the human data. The correlation shifts from 0.997 to 0.998 and reduced the mean square error (MSE).

[1] https://sites.psu.edu/ftehranchi/projects/

Table 1: The mean task completion time in seconds for the four learning sessions for the Dismal mouse-interface task (Paik et al., 2015) and correlation with the human data (N=30).

Day	Human		Original Model		JSegMan Correction Hands		JSegMan Correction Hands and Eyes	
	M	SE	M	SE	M	SE	M	SE
1	1366	60.8	1326	12.078	1338	12.06	1339	11.72
2	894	26.6	891	6.175	893	5.144	894	6.498
3	727	25.5	693	4.496	700	6.207	704	5.019
4	659	22.7	594	5.775	603	4.35	614	4.381
Correlation			0.997		0.9978		0.9984	
MSE			1747.5		1162.5		820.75	

Discussion and Conclusion

In this paper, we described our efforts to bridge the gap of interaction between cognitive models and task environments. This article focused on models written with the ACT-R cognitive architecture, but other architectures could use this approach and system.

We call our approach JSegMan because simulation eyes and hands in ACT-R models require the original ACT-R model, the Robot and Sikuli packages in Java, and currently Emacs (as a connector). The JSegMan approach will increase the usability, applicability, and accessibility of cognitive architectures. Also, it can be used as a cognitive model examiner—to see if the knowledge can do the task.

Using the JSegMan eyes and hands simulation in place of a user, questions about user interface designs such as evaluating designs, changing the interface and examining the effects on task performance can be answered more efficiently. The proposed approach can prove the advantages of CMIMS in HCI, and realize the use of user models in system design (Pew & Mavor, 2007). JSegMan can help implement this approach by testing user interfaces, making this process more approachable and more practical. Our results running the Dismal model and finding missing knowledge illustrates that JSegMan also offers the ability to understand a model more accurately.

The advantages of this approach are: (a) ACT-R code does not change due to the JSegMan, (b) JSegMan does not affect the ACT-R response time because JSegMan has separate timing formula and runs along with ACT-R, and (c) models are able to interact with any interface on a PC.

Further Research and Limitations

Further work remains. We still need to manually check if the action takes place correctly in JSegMan and the eyes follow the hands successfully, as well as exploring error generation and correction. Furthermore, we plan to use an object recognition algorithm to extract the visible objects without pre-defining them for models, but this functionality is beyond current ACT-R's action execution.

JSegMan should use the OCR capability more directly as it is likely more efficient and more comfortable to use. Additionally, the nested pattern matching that follows the EPIC visual search will be useful to implement. The current screen scanning approach is a bit mechanistic. The scan starts in the upper left every time. It is probably more realistic to start either where the previous task left off or based on other heuristics that people use (Hornof & Kieras, 1997).

In the future, we plan on offering an installation method that includes bundled versions of all dependencies, allowing near plug and play support with ACT-R. JSegMan components could also be expanded so JSegMan can observe the users, collect more realistic inputs, and thus better predict human performance. Therefore, JSegMan can be a substitute for humans in the software testing process and can be considered as a software testing tool.

Acknowledgments

This work was funded partially by ONR (N00014-15-1-2275). David Reitter provided useful comments on Emacs and Aquamacs (the Emacs version for Mac). We wish to thank Jong Kim who provided the idea for ESegMan and Dan Bothell for his assistance with ACT-R.

References

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

Anderson, J. R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science, 8*, 87-129.

Anderson, J. R., Matessa, M., & Lebiere, C. (1998). The visual interface. In J. R. Anderson & C. Lebiere (Eds.), *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Bradski, G. B. (2004). *Open source computer vision library*: Springer.

Burns, M., Ritter, F. E., & Zhang, X. (2016). Using Naturalistic Typing to Update Architecture Typing Constants. In *Proceedings of ICCM - 2016-14th International Conference on Cognitive Modeling*. University Park, PA: Penn State.

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.

Byrne, M. D., & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebiere (Eds.), *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Byrne, M. D., & Kirlik, A. (2005). Using computational cognitive modeling to diagnose possible sources of aviation error. *International Journal of Aviation Psychology, 15*(2), 135-155.

Byrne, M. D., Wood, S. D., Sukaviriya, P., Foley, J. D., & Kieras, D. E. (1994). Automating interface evaluation. In *Proceedings of the CHI'94 Conference on Human Factors in Computer Systems*, 232-237. ACM: New York, NY.

Fu, D., Houlette, R., Jensen, R., Bascara, O., & San Mateo, C. (2003). A visual, object-oriented approach to simulation behavior authoring. In *Proceedings of the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC 2003)*.

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.

Hornof, A. J., & Kieras, D. E. (1997). Cognitive modeling reveals menu search is both random and systematic. In *Proceedings of the CHI'97 Conference on Human Factors in Computer Systems*, 107-114. ACM: New York, NY.

Kasper, M., Correll, N., & Yeh, T. (2014). Abstracting perception and manipulation in end-user robot programming using Sikuli. In *Technologies for Practical Robot Applications (TePRA), 2014 IEEE International Conference on*, 1-6. IEEE.

Kieras, D. E. (2009). Model-based evaluation. *The human-computer interaction: Development process*, 294-310.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12*, 391-438.

Kuchhal, P. (2014). Ameliorating the image matching algorithm of Sikuli using Artificial Neural Networks. *International Journal of Computer Science & Communication, 5*(1), 1-4.

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Lohse, G. L. (1997). Models of graphical perception. In M. Helander, T. K. Landauer & P. Prabhu (Eds.), *Handbook of Human-Computer Interaction* (pp. 107-135). Amsterdam: Elsevier Science B. V.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Paik, J., Kim, J. W., Ritter, F. E., & Reitter, D. (2015). Predicting user performance and learning in human-computer interaction with the Herbal compiler. *ACM Transactions on Computer-Human Interaction, 22*(5), Article No.: 25.

Pew, R. W., & Mavor, A. S. (Eds.). (2007). *Human-system integration in the system development process: A new look*. Washington, DC: National Academy Press.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction, 7*(2), 141-173.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2001). User interface evaluation: How cognitive models can help. In J. Carroll (Ed.), *Human-Computer Interaction in the New Millenium* (pp. 125-147). Reading, MA: Addison-Wesley.

Ritter, F. E., Kukreja, U., & St. Amant, R. (2007). Including a model of visual processing with a cognitive architecture to model a simple teleoperation task. *Journal of Cognitive Engineering and Decision Making, 1*(2), 121-147.

Ritter, F. E., & Wood, A. B. (2005). Dismal: A spreadsheet for sequential data analysis and HCI experimentation. *Behavior Research Methods, 37*(1), 71-81.

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. *Cognitive Systems Research, 1*(4), 201-220.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors, 48*(3), 362-380.

Salvucci, D. D. (2009). Rapid prototyping and evaluation of in-vehicle interfaces. *ACM Transactions on Computer-Human Interaction, 16*(2), Article 9, 33 pages.

Salvucci, D. D. (2013). Integration and reuse in cognitive skill acquisition. *Cognitive Science, 37*(5), 829-860.

Soegaard, M. (2013). Interaction design foundation. *Interaction Design Foundation–Free educational materials*.

Soloway, W. L. J.-E., & Johnson, W. (1984). Intention-based diagnosis of programming errors. In *Proceedings of the 5th National Conference on Artificial Intelligence, Austin, TX*, 162-168.

St. Amant, R., Riedel, 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. Erlbaum: Mahwah, NJ.

St. Amant, R., & Riedl, M. O. (2001). A perception/action substrate for cognitive modeling in HCI. *International Journal of Human-Computer Studies, 55*(1), 15-39.

Tehranchi, F., & Ritter, F. E. (2017). An eyes and hands model for cognitive architectures to interact with user interfaces. In *MAICS, The 28th Modern Artificial Intelligence and Cognitive Science Conference*, 15-20. Fort Wayne, IN: Purdue University.

Tehranchi, F., & Ritter, F. E. (2018). Using Java to Provide Cognitive Models with a Universal Way to Interact with Graphic Interfaces. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation.* Washington DC, USA.

Yeh, T., Chang, T.-H., & Miller, R. C. (2009). Sikuli: Using GUI screenshots for search and automation. In *Proceedings of the 22nd Annual ACM symposium on User interface software and technology*, 183-192. ACM.

Modelling the Effect of Time-on-Task Fatigue in Prolonged Driving

Leong-Hwee Teo (tleonghw@dso.org.sg)
DSO National Laboratories, 12 Science Park Drive, Singapore 118225

Grace W.-Y. Ang (awanyu@dso.org.sg)
DSO National Laboratories, 12 Science Park Drive, Singapore 118225

Abstract

Integrated models of psychomotor vigilance test (PVT) and driver behavior with fatigue mechanism were used to model, at an individual level, the effect of time-on-task fatigue on driving performance over a 4-hour driving duration. PVT data collected from 25 participants in a first driving trial was used to fit model parameters to individuals' PVT performance over time. The individualized parameters were then used in the driver behavior model to predict driving performance over time in a second driving trial.

Keywords: Fatigue; Time-on-Task; Driving; Individual Differences; Computational Model; ACT-R

Introduction

The effect of fatigue on task performance is of concern to organizations and individuals who may be routinely required to perform their work for prolong periods of time. Such sustained operations, often compounded with sleep deprivation, is a challenge faced by military personnel, fire-fighters and other emergency services responders.

While current ways to manage fatigue may involve pre-work screening for signs of fatigue, or real-time monitoring of eye-closure to detect the onset of a sleep attack, we believe improving the planning and assignment of individuals to more precisely meet the nature and duration of upcoming work sessions, will contribute to a more comprehensive solution. In transport operations, it is convenient to assign the driver with the most amount of rest or the least amount of hours worked. However, if the pool of drivers become constrained, it may be better to assign a less rested driver that is still expected to stay alert enough to drive safely for the duration of the upcoming trip, and keep a fresher driver in reserve to meet a potential demand from a longer duration trip.

This paper reports our research towards such a predictive tool. Our modelling and evaluation extended the work by Khosroshahi et al. (2016) by modelling the effect of fatigue on task performance at the individual level, focusing on time-on-task fatigue over a 4-hour driving duration and providing further validation of Khosroshahi's models against another human dataset from a previously completed study on prolonged driving.

Models of PVT and Driving Under Fatigue

We used the integrated models of psychomotor vigilance test (PVT) and driver behavior with fatigue mechanism developed by Khosroshahi at al. (2016), which were based on the fatigue mechanism and PVT model developed by Gunzelmann and colleagues (Walsh, Gunzelmann and Van Dongen, 2017; Veksler and Gunzelmann, 2017) and the model of driver behavior by Salvucci (2006), which were all implemented in the ACT-R cognitive architecture (Anderson, 2007). Here, we briefly describe the integrated models.

In ACT-R, the procedural knowledge to perform PVT, driving, or any modelled task, is represented by action-selection rules that specify conditions about the current cognitive state that must be satisfied for the rule to execute its actions. In each cognitive cycle, from amongst all matching action-selection rules, the rule with the highest utility value and that exceeds the utility threshold, will be selected and executed. Gunzelmann's fatigue mechanism attenuates the utility values of action-selection rules by incorporating a multiplier ***FP***, based on prior sleep, time awake, time-of-day and time-on-task, to the initial utility ***IU***, of the action-selection rule:

$$Utility = IU * FP + noise$$

$$FP = FPpercent * (1 - FPBMC * biomath) * (1 + time\text{-}on\text{-}task)^{FPMC}$$

The influence of prior sleep, time awake and time-of-day is captured in ***biomath***, an alertness index computed by a biomathematical model of fatigue (McCauley et al., 2013). ***FPpercent*** captures the cumulative effect of *microlapses*, where if in a cognitive cycle no action-selection rule exceeds the utility threshold, the model in effect delays for the duration of that cognitive cycle and "does nothing". Every time a microlapse happens, ***FPpercent*** is decremented by a small value, which not only attenuates utility values, it also increases the likelihood of a subsequent microlapse. ***FPBMC*** and ***FPMC*** are regression coefficients that scale the fatigue effects arising from ***biomath*** and ***time-on-task*** respectively.

The fatigue mechanism also attenuates the utility threshold ***UT***, to increase the likelihood that utility values of matching action-selection rules exceed the threshold and avoid the occurrence of a microlapse:

$$UT = UT_0 * (1 - UTBMC * biomath) * (1 + time\text{-}on\text{-}task)^{UTMC}$$

UT_0 is the initial utility threshold value, while ***UTBMC*** and ***UTMC*** are regression coefficients that scale the fatigue effects arising from ***biomath*** and ***time-on-task*** respectively.

Attenuating the utility threshold represents a compensatory mechanism to offset the increasing influence of fatigue, but this also makes rule selection less stringent.

The PVT and driver behavior models contain their respective action-selection rules that perform simulated PVTs and driver behavior in a driving simulation. When these models are executed in an ACT-R environment with the fatigue mechanism enabled, attenuations to utility values and the utility threshold result in the occurrence of microlapses, which lead to increasing momentary delays in task execution, as a manifestation of the negative impact from fatigue on task performance.

Human Data from Driving Study

To fit model parameters and test model predictions, we used data from a previous unpublished study on prolonged driving in an actual driving circuit done in 2011. Each participant underwent two different driving protocols, counterbalanced and separated by 6 days. In protocol A, participants drove for 45 minutes, then stopped and performed a 15-minute cognitive test battery which included a 5-minute Psychomotor Vigilance Test (PVT), and resumed driving, repeating this cycle 4 times. In protocol B, participants drove continuously for 4 hours and performed the test battery after driving. In both protocols, participants performed the same 15-minute test battery before the start of each protocol. Participants drove on road at 30km/h in a quadrilateral circuit of approximately 170m for each straight stretch of road, with no other vehicles or obstacles present. If the participant committed a lane deviation before the end of the protocol, the participant would stop driving, perform the test battery, and the driving trial ends.

Of the 40 participants in the driving study, 11 participants successfully completed both protocols without committing any lane deviations; they were classified as *resilient* drivers. 14 participants failed to complete both protocols having committed a lane deviation in each trial before the end of the protocol; they were classified as *vulnerable* drivers. The rest of the 15 participants completed one of the protocols but failed to complete the other. As the original study suspected that these participants may had intentionally committed the half-lane deviations, we decided to use only the data from the resilient and vulnerable groups of drivers.

Psychomotor Vigilance Test (PVT)

The PVT is an established test of a person's level of fatigue (Basner et al., 2018). The test taker is to respond as quickly as possible, by pressing a button, upon seeing the appearance of a visual cue, which occurs at random inter-stimulus intervals of 2 to 10 seconds.

Figure 1 plots the PVT median reaction times by each driver over the course of the driving protocols. For the vulnerable drivers, data points that lie between the scheduled tests are for PVTs administered after a lane deviation was committed, which ended the driving trial. We can see there were individual differences in both baseline performance and susceptibility to time-on-task fatigue.

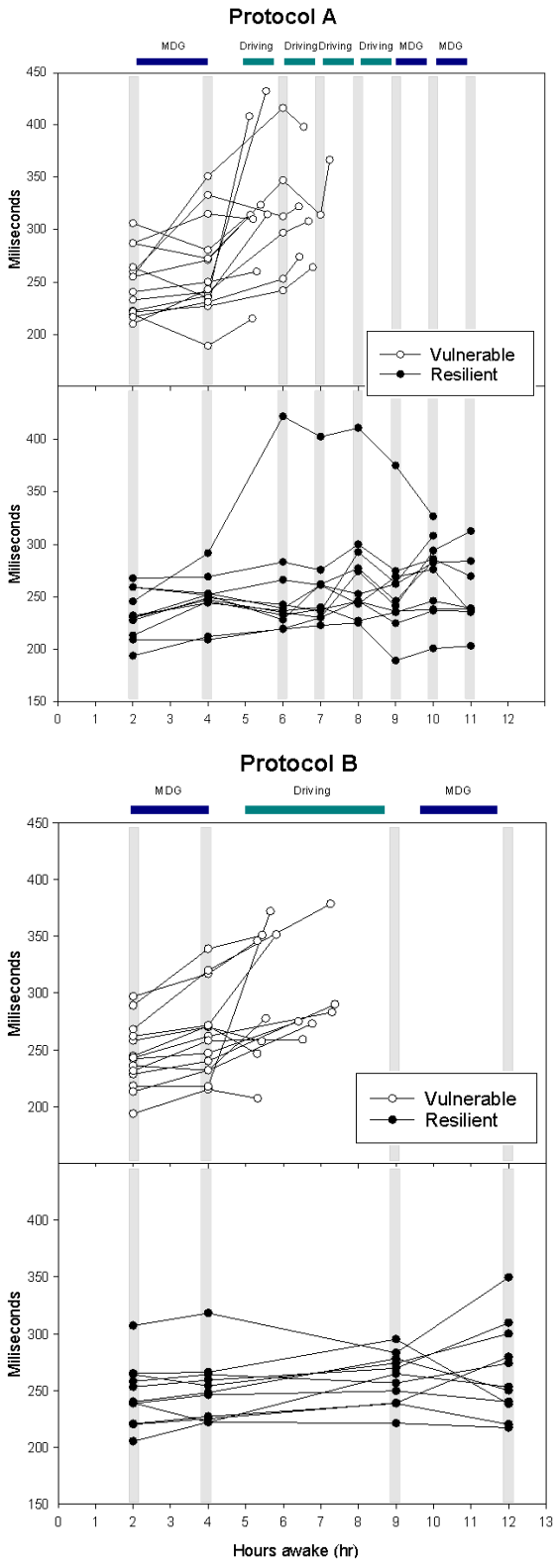


Figure 1: PVT median reaction times in the study protocols. PVTs administered between the 4th to 9th hour (both inclusive) were part of the driving protocol. Vulnerable drivers committed a lane deviation that ended their driving trial. "MDG" was a driving simulation game.

Driving Performance

We also used two measures of driving performance from the driving study data. First, was the time-point the driver committed a lane deviation, or none for the resilient drivers. The second measure was standard deviation of lateral position (Verster and Roth, 2011), which we extracted through automated processing of videos taken from a forward-facing camera mounted behind the vehicle's windscreen. The gray lines in Figure 3 plot the standard deviation of lateral position (SDLP) by resilient drivers, where a higher SDLP indicated poorer driving performance: more "weaving" as the driver tried to drive straight and stay in lane. The SDLP for vulnerable drivers are not shown due to constraint of space.

Modelling Results

In protocol A, participants did 4 rounds of 45-minute driving, interspaced with 3 PVTs, and a PVT each before and after the driving protocol. We made a simplifying assumption that changes in PVT performance across these 5 PVTs were due to the time-on-task fatigue effect arising from the main driving activity. There was also the fatigue effect arising from prior sleep, time awake and time-of-day, although all participants had similar amount of rest based on 7 days of Actiwatch data and similar driving start times based on the study protocol.

We first used the PVT model with fatigue mechanism set to perform a 4-hour PVT session. By varying model parameters to match the predictions of PVT performance at the above 5 time segments to the actual PVT performance in the corresponding 5 PVTs administered in the driving protocol, we got time-on-task fatigue mechanism parameters ***FPMC*** and ***UTMC***, to reflect the time-on-task fatigue effect arising from the driving activity. We also varied ***IU***, ***UT$_0$***, and the ACT-R cognitive cycle time ***dat***, to reflect individual differences. The cognitive cycle time was used by Gunzelmann et al. (2009) to capture individual differences in PVT performance.

To obtain an individualized set of fatigue mechanism parameters and cognitive cycle time, we repeated the model fitting for each participant's data. We then used each individualized set of fatigue mechanism parameters and cognitive cycle time in the driver behavior model with fatigue mechanism, to generate predictions of driving performance following the continuous 4 hours of driving in protocol B. This transfer of model parameters from the PVT model to the driving behavior model was used in Khosroshahi at al. (2016). Here, we applied this approach to time-on-task fatigue over a 4-hour driving duration, and at the individual level.

Resilient Drivers

Figure 2 plots the median response time (MRT) by drivers at each of the five PVTs in driving protocol A and the predicted MRT given the fatigue mechanism parameters and cognitive cycle time for that particular driver. The combination of these fatigue mechanism parameters and cognitive cycle time was able to capture the difference in individual susceptibility to the effects of time-on-task fatigue due to the driving activity.

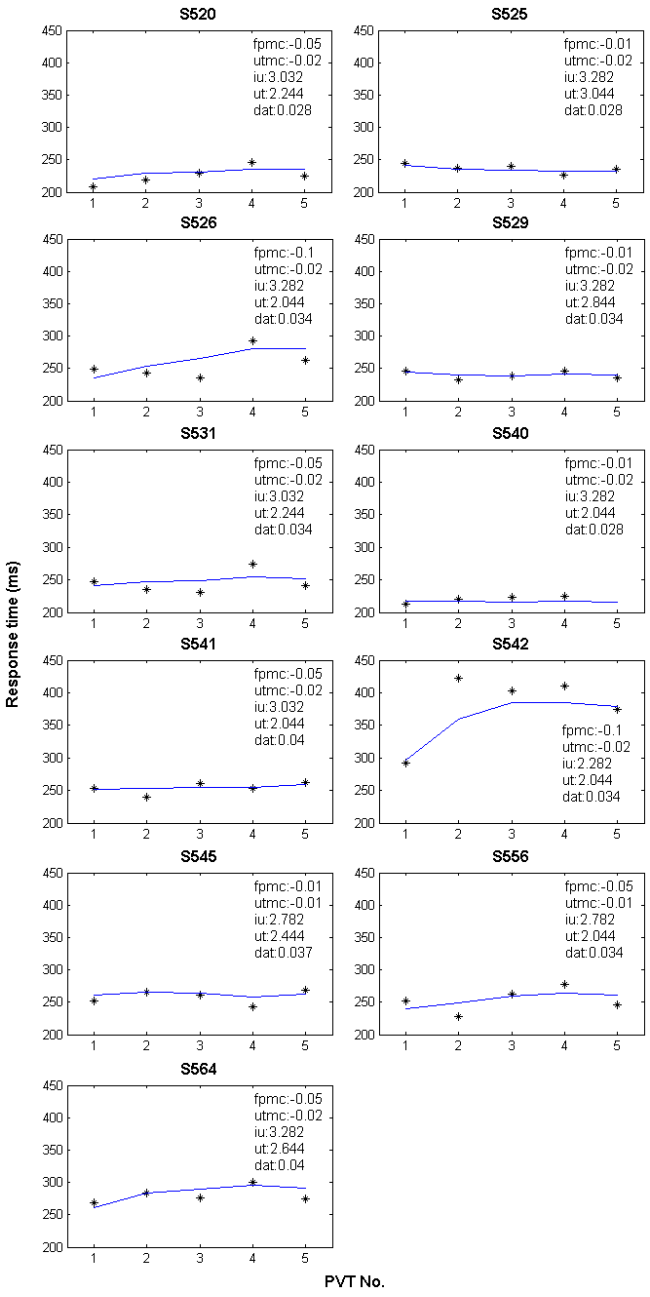


Figure 2: Individualized fits of PVT for resilient drivers. Asterisks are the median response time (MRT) by drivers at each of the five PVTs in driving protocol A. The lines plot the predicted MRT given the model parameters for that particular driver.

We then used the same individualized fatigue mechanism parameters and cognitive cycle time in the driver behavior model with fatigue mechanism, to predict how each driver would perform in 4 hours of continuous driving in protocol B. Figure 3 plots the predictions of standard deviation of lateral position (SDLP) for each resilient driver.

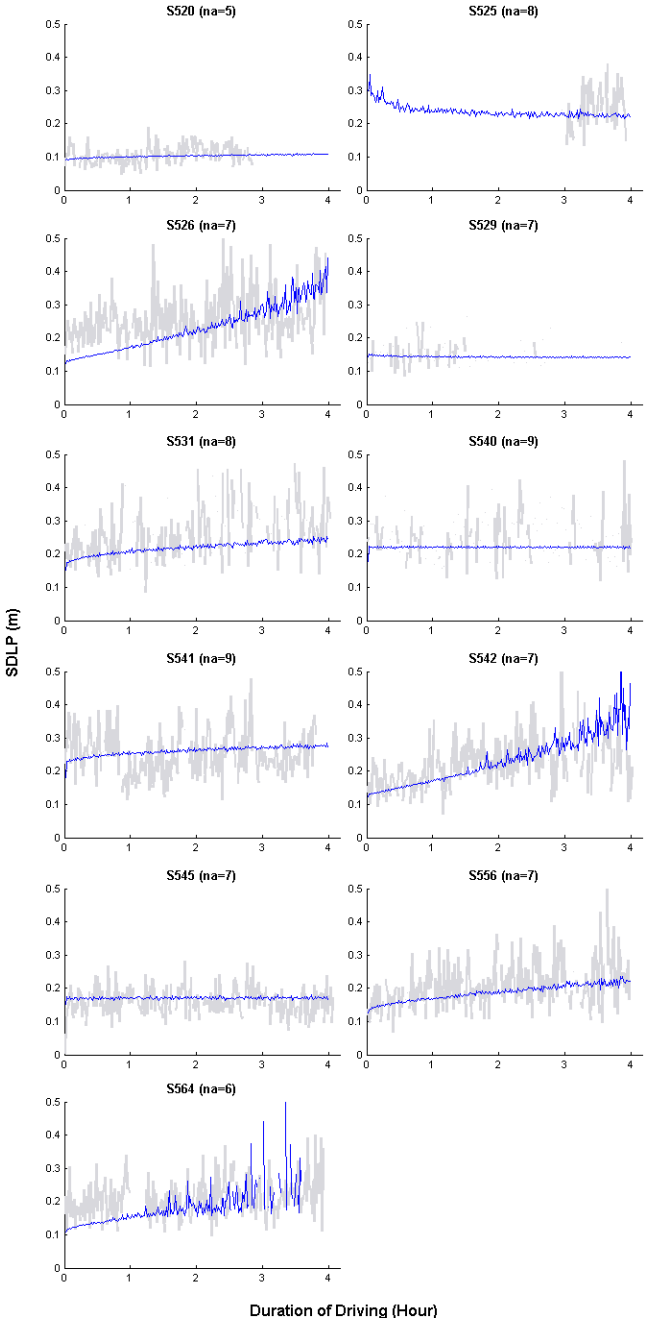


Figure 3: Individualized predictions of standard deviation of lateral position (SDLP) for resilient drivers. The gray plots are the actual performance in driving protocol B; gaps in the gray plots are due to failure to extract lane position because of poor visibility of lane markers in the videos. The blue plots are predicted SDLP given the individualized model parameters for each driver.

To achieve these fits between predicted and actual SDLP, another parameter in the driver behavior model (parameter k_l in Salvucci, 2006; coded as variable ***na*** in the model) was varied for each driver to scale the predicted SDLP values to align with the SDLP values exhibited by each driver in the driving trials. Salvucci (2006) explained that the driver behavior model "involves a number of domain-specific parameters, some of which may vary among individual drivers", and "these parameters were estimated by setting them to reasonable values, observing the resulting qualitative and quantitative fits given these values, and revising the values accordingly". In particular, the variable *na* was described by Salvucci (2004) to affect how "the model continually adjust steering to maintain a near point centered on the roadway", which may reflect individual differences in driving style and how aggressively each driver tried to continuously minimize the distance between the center of the vehicle and the center of the driving lane.

Figure 4 plots the proportion of simulated driving trials which the driver behavior model committed a lane deviation in each 1-minute interval of driving duration. Since resilient drivers completed the full 4-hour driving duration, the desired prediction would be zero or low probability of lane deviation for the entire 4-hour duration. If we used <=20% as the criteria for low probability of predicted lane deviation, the driver behavior model with fatigue mechanism would advise that 7 of the 11 resilient drivers would be able to complete driving protocol B.

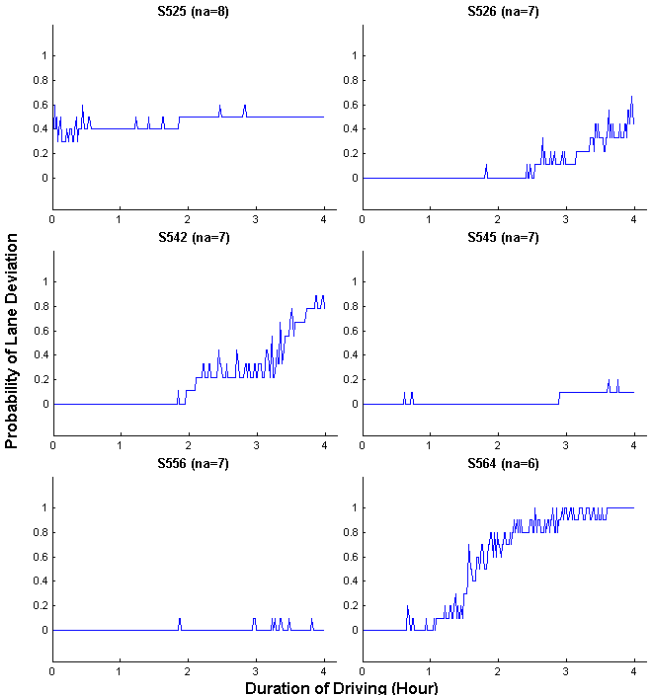


Figure 4: Individualized predictions of probability of lane deviation for each 1-minute interval of driving by resilient drivers. The model predicted 5 out of the 11 resilient drivers have zero probability of lane deviation. Shown here are the 6 resilient drivers with non-zero probabilities.

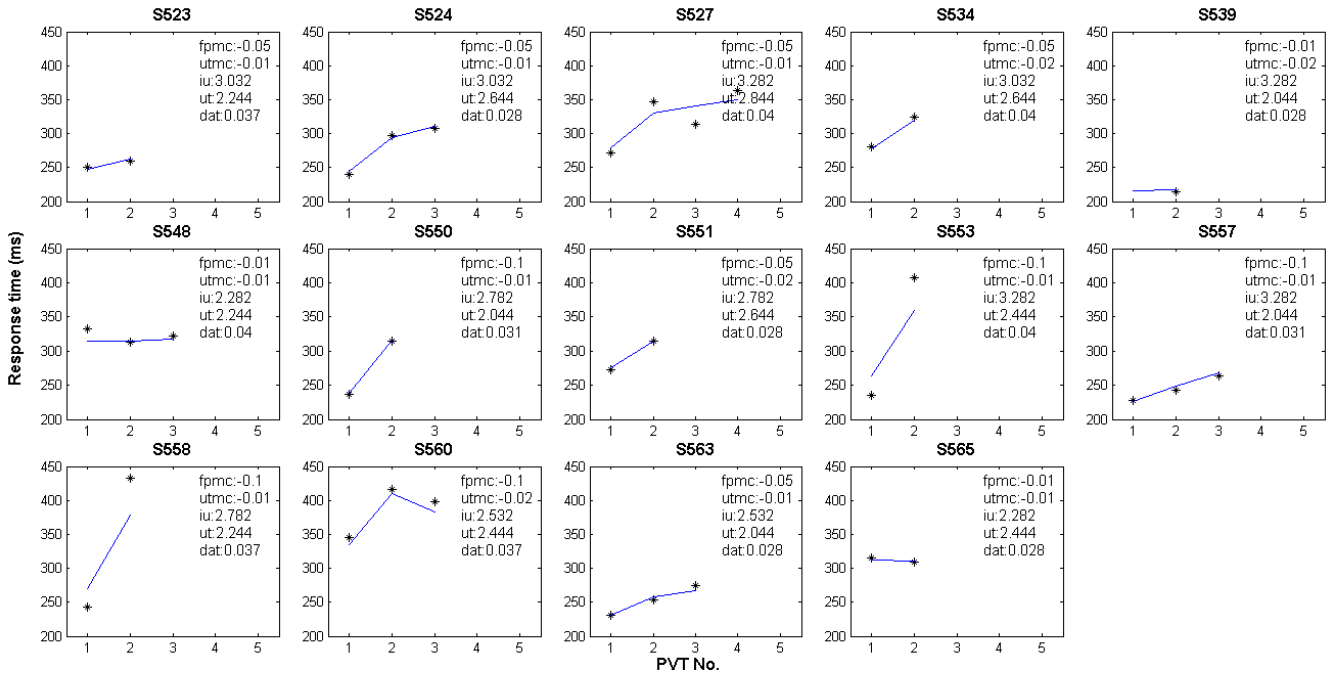


Figure 5: Individualized fits of PVT for vulnerable drivers. Asterisks are the median response time (MRT) by drivers in the PVTs in driving protocol A, where the last PVT administered was after they committed a lane deviation. The lines plot the predicted MRT given the model parameters for that particular driver.

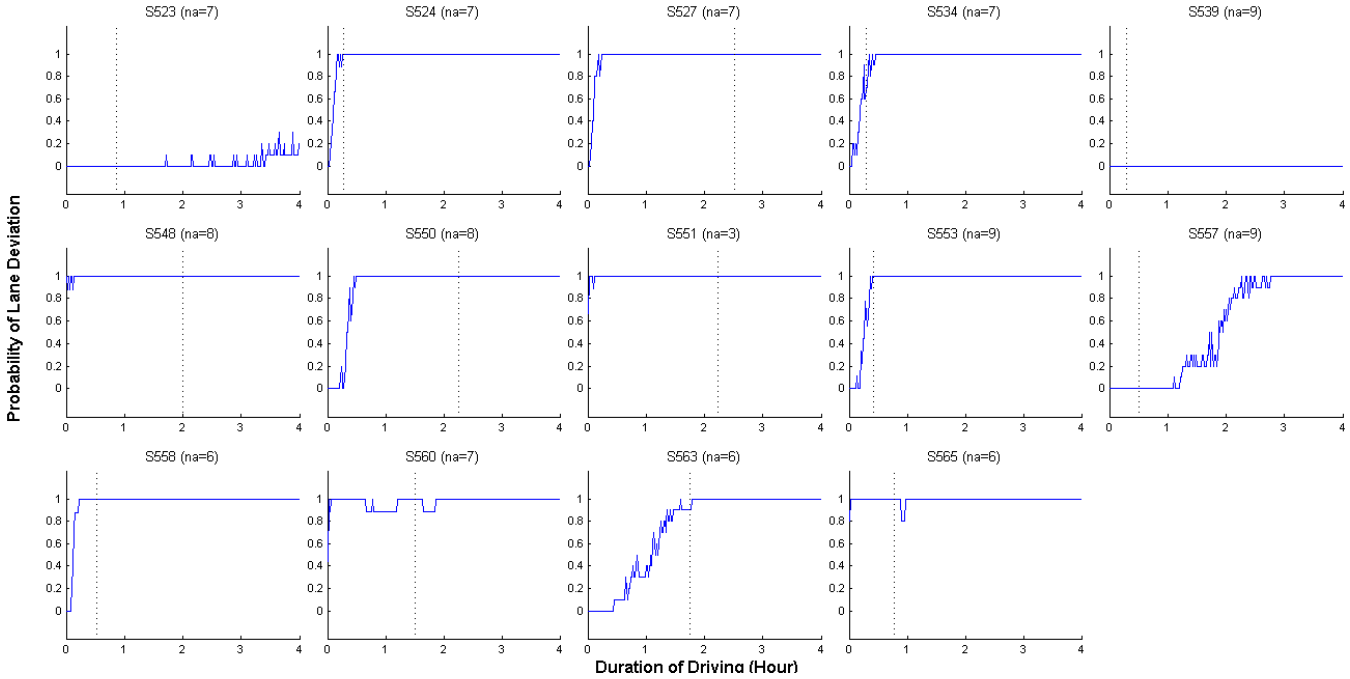


Figure 6: Individualized predictions of probability of lane deviation for vulnerable drivers. The solid lines plot the predicted probability of lane deviation for each 1-minute interval of driving duration, given the specific model parameters for the individual driver. The dotted vertical lines mark when each driver committed a lane deviation in protocol B.

Vulnerable Drivers

We performed the same fitting of fatigue mechanism parameters and cognitive cycle time for the vulnerable drivers. Figure 5 plots the median response time (MRT) by vulnerable drivers for the PVTs they performed in driving protocol A, before and after they committed the lane deviation, and the predicted MRT given the fatigue mechanism parameters and cognitive cycle time for that particular driver. We then used the same individualized fatigue mechanism parameters and cognitive cycle time in the driver behavior model with fatigue mechanism, to predict how each driver would perform in 4 hours of continuous driving in protocol B.

Figure 6 plots the proportion of simulated driving trials which the driver model committed a lane deviation in each 1-minute interval of driving duration. Since vulnerable drivers did commit a lane deviation in the driving trials, the desired prediction would be an elevated probability of lane deviation at when the driver actually committed a lane deviation in protocol B. If we used >20% as the criteria for elevated probability of predicted lane deviation, then the driver behavior model with fatigue mechanism would advise that 5 of the 14 vulnerable drivers (S524, S534, S553, S558 and S565) would have elevated probability of lane deviation in the same hour-segment as when the driver actually committed a lane deviation in driving protocol B.

The model would also conservatively advise that 6 other vulnerable drivers (S527, S548, S550, S551, S560 and S563) would have elevated probability of lane deviation in an hour-segment before when the driver actually committed a lane deviation in driving protocol B. However, the model will advise that 2 other vulnerable drivers (S523 and S557) would have elevated probability of lane deviation in an hour-segment after when the driver actually committed a lane deviation, and the other 1 vulnerable driver (S539) to have zero probability of lane deviation in 4 hours of continuous driving in protocol B.

Discussion

We had two possible explanations for the difference between model predictions and actual driving performance. The first was there was only moderate correlation between PVT performance and driving performance (as measured by SDLP in the last 15 minutes of driving just before the PVT) by study participants in both protocol A ($r=0.513$) and protocol B ($r=0.583$). This meant that for this dataset, PVT was not particularly predictive of driving performance. For example, vulnerable driver S539's PVT performance would had suggested good resilience to fatigue, but nonetheless committed a lane deviation within the first hour of driving. Another possible reason might be that some resilient drivers had $\boldsymbol{IU}$ and $\boldsymbol{UT_0}$ parameters values that were too close to each other. For example in resilient driver S525, with noise in the utility values computation, the close $\boldsymbol{IU}$ and $\boldsymbol{UT_0}$ values would had resulted in more occurrences of microlapses that impacted predicted driving performance.

The dataset we used in this study was limited in both the number of participants (n=25) and the number of PVT sessions (as few as 2 for vulnerable drivers and 5 for resilient drivers) to fit model parameters. For future work, we hope to collect or obtain datasets that have more data points for each participant, to see if that will lead to more predictive model parameters for each participant. We also hope to collect or obtain more datasets on prolonged driving performance with different driving protocols, to evaluate and refine the method of fitting model parameters.

Acknowledgments

The authors will like to thank Dr. Frederick Tey and his team for sharing the driving study data, Dr. Glenn Gunzelmann and Dr. Bella Veksler for sharing the ACT-R fatigue mechanism and PVT model, and Professor Dario Salvucci and Ehsan Khosroshahi for sharing the integrated fatigue, PVT and driver behavior models in Java ACT-R.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Basner, M., Hermosillo, E, Nasrini, J., McGuire, S., Saxena, S., Moore, T. M., Gur, R. C., & Dinges D. F. (2018). Repeated Administration Effects on Psychomotor Vigilance Test Performance. *Sleep, 41*(1), zsx187.

Gunzelmann, G., Moore, L. R., Gluck, K. A., Van Dongen, H. P. A., & Dinges, D. F. (2009). Examining sources of individual variation in sustained attention. In *Proceedings of the 31st Annual Meeting of the Cognitive Science Society* (pp. 608–613). Austin, TX: Cognitive Science Society.

Khosroshahi, E. B., Salvucci, D. D., Veksler, B. Z., & Gunzelmann, G. (2016). Capturing the Effects of Moderate Fatigue on Driver Performance. In *Proceedings of the 14th International Conference on Cognitive Modeling* (pp.163-168).

McCauley, P., Kalachev, L. V., Mollicone, D. J., Banks, S., Dinges, D. F., & Van Dongen, H. P. A. (2013). Dynamic Circadian Modulation in a Biomathematical Model for the Effects of Sleep and Sleep Loss on Waking Neurobehavioral Performance. *Sleep, 36*(12), 1987–1997.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors, 48*, 362-380.

Veksler, B. Z. & Gunzelmann, G. (2018). Functional Equivalence of Sleep Loss and Time on Task Effects in Sustained Attention. *Cognitive Science, 42*, 600-632.

Verster, J. C., & Roth, T. (2011). Standard operation procedures for conducting the on-the-road driving test, and measurement of the standard deviation of lateral position (SDLP). *International Journal of General Medicine, 4*, 359–371.

Walsh, M. M., Gunzelmann, G. & Van Dongen, H. P. A. (2017). Computational cognitive modeling of the temporal dynamics of fatigue from sleep loss. *Psychonomic Bulletin & Review, 24*, 1785–1807.

Analysis of Learning Action Selection Parameters in a Neural Cognitive Model

Sverrir Thorgeirsson (sverrir.thorgeirsson@uwaterloo.ca)
Terrence C. Stewart (tcstewar@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo 200 University Avenue West,
Waterloo, ON, Canada, N2L 3G1

Abstract

In our previous work, we have implemented a biologically realistic action selection system that can perform complex tasks such as sentence parsing, the n-Back task and the Tower of Hanoi. Although our models have successfully performed those tasks, they have so far required human researchers to tune multiple parameters before the models can be expected to exhibit good performance. In this paper, we show that an improved, parameter-sparse learning rule can be applied to a cognitive sequencing task.

Keywords: neural engineering framework; spiking neurons; computational neuroscience; neural production systems; basal ganglia

Introduction

Our previous research has been centred on the construction of spiking neural models that can perform diverse and complex cognitive tasks. In recent years, we have introduced models of the Tower of Hanoi (Stewart & Eliasmith, 2011), bandit tasks (Stewart, Bekolay, & Eliasmith, 2012), command parsing (Stewart & Eliasmith, 2013), sentence parsing (Stewart, Choo, & Eliasmith, 2014), the n-Back task (Gosmann & Eliasmith, 2015), action planning (Blouw, Eliasmith, & Tripp, 2016), speech production (Kröger, Bekolay, & Blouw, 2016) and the effects of reduction of dopamine on speech production (Senft et al., 2016), hierarchical reinforcement learning of navigation and abstract action rules (Rasmussen, Voelker, & Eliasmith, 2017), and semantic memory search (Kajic et al., 2017).

Our work uses the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003) and the Semantic Pointer Architecture (SPA; Eliasmith, 2013) which are general methods that can be used to investigate cognitive theories using detailed models of spiking neurons. Our intent is to apply knowledge of higher-level cognitive behaviour to explore and test theories and make predictions of low-level phenomena in the brain such as neural spiking patterns and connectivity. In this way, our research should serve as a connection between cognitive science and neuroscience (Stewart & Eliasmith, 2011; Eliasmith, 2013). It is for this reason that we use biologically realistic methods (i.e., the SPA) which can be thought of as neural approximations of cognitive production systems.

However, this approach comes with the cost of lower algorithmic expressibility, as crucial programming rules such as IF-ELSE statements are difficult to express without the time-consuming process of parameter tuning. In a recent work (Stewart, Thorgeirsson, & Eliasmith, in press), we introduced a solution to this problem with an automatic learning rule so that our models can learn their parameters autonomously. In this paper, we apply this rule on alphabetical sequencing tasks of varying sizes and complexities. Our results show that the new approach works successfully for any task size that we tested, with the time that the model spends learning growing proportionally with the size and scope of the task.

The Neural Engineering Framework

The NEF provides us a way to represent and transform information encoded in neurons. That way, we can generate biologically realistic models such as the one presented in this paper.

The three principles of the NEF are those of representation, transformation and dynamics (Eliasmith & Anderson, 2003):

1. Representation: To determine how a group of neurons can represent a high-dimensional vector, the NEF uses distributed representations. Every neuron in the group has a direction in space in which it will fire mostly strongly. This direction is called its preferred direction vector **e**. Under the NEF, we claim that the input current to a neuron is a linear function of the value to be represented, x. So, for a function G, gain parameter α, and constant background bias b, we can compute the neural activity given x with this equation:

   $$a = G(\alpha e \cdot x + b)$$

   The NEF assumes that this encoding of x can be decoded linearly to specify the neural representation of x.

   For the model presented in this paper, we use spiking Leaky-Integrate-and-Fire (LIF) neurons for the function G, which has the advantage of both capturing a high level of biological detail while also being computationally efficient.

2. Transformation: The NEF can be used to solve the synaptic connection weights between populations analytically. Computations can be performed on represented vectors through various choices of decoders. This means that for two neuron populations A and B, the connection between those populations can approximate the function $f(x)$, where x is some vector that is represented by A.

3. Dynamics: The NEF is capable of solving equations of the form $dx/dt = A(x) + B(u)$ where x is a value that is being represented by a neural population, u is any input and A and B are arbitrary functions. In this paper, we use a special

case of this equation where $B(u) = u$ and $A(x) = 0$ (i.e., when there is no input to the system it preserves the current state of the visual system of our model).

For many models, including the one introduced in this paper, it is important to compute the similarity between the current state x and some other state s. To that end, we will simply compute the dot product of x and s neurally which gives us a single number that represents that similarity.

Neural Action Selection

Figure 1 illustrates the core action selection system in SPA which we will use in this paper. This model approximates the anatomy, connectivity and neural properties of the basal ganglia, thalamus and cortex (Stewart & Eliasmith, 2011) and can be used in a general-purpose way to model different action selection systems. The overall model consists of approximately 6000 leaky integrate-and-fire (LIF) spiking neurons. For further details of the construction of this model, see (Stewart, Thorgeirsson, & Eliasmith, in press).

The cortex represents cognitive state in our model. In our case, it contains of the contents of the visual system and the working memory. The connections between the cortex and the striatum in the basal ganglia determine the utility of each action. The basal ganglia determines the largest of those utility values. The connections via the thalamus execute the action that updates the cortical state.

To skip the time-consuming and potentially unreliable stage of optimizing the scaling factors in the utility functions, we previously proposed the use of a simple online learning rule that constantly updates the scaling factors while the model is running:

$$\Delta\omega_{i,j} = \alpha x_i (t_j - y_j)$$

(Widrow & Hoff, 1960). This is meant for situations where the adjustable parameters $\omega_i j$ are weights on x_i that produce a weighted sum y_j (i.e. $y_j = \sum_i (x_i \omega_{ij})$). In our case, x_i represents the neural activity in the cortical neurons that compute the utility of action i, y_i gives the represented output of the neural system and t_j is a training signal.

Learning Task

We chose to test our approach on a cognitive task in which the goal is for the working memory to recite the letters in n lists with m letters each. We call those lists alphabets. The name of the alphabet that should be recited is given in the visual system, which cycles through all available alphabets periodically.

For $n = m = 5$, our alphabets are shown in Table 1.

This means that if the visual system contains ALPHA, then the working memory should cycle through $P \to K \to U \to J \to M \to P \to \ldots$. For $n, m < 5$, we consider the first m letters in the first n alphabets.

The model can be implemented as a production system with the production rules

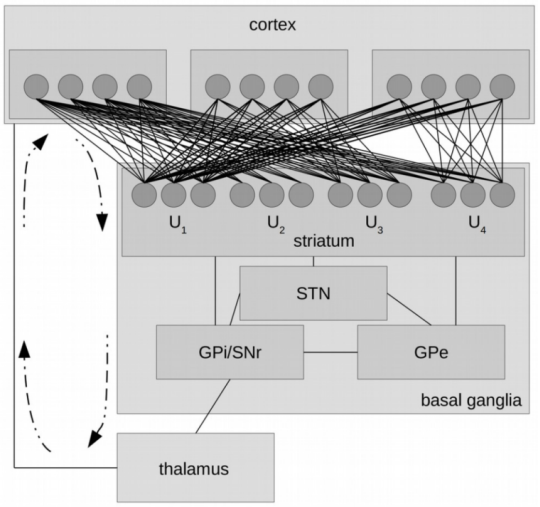


Figure 1: The cortex-basal ganglia-thalamus loop that forms the neural action selection and execution system. Neurons (dark circles) and connections are shown only for the inputs to the action selection system. These connections compute the utility (Ui) of each of the actions i, given the current cortex state. The basal ganglia selects the action with the highest utility, and the thalamus executes that action.

IF	VISION = A
AND	STATE = L
THEN	STATE = N

for each pair (L, N) in each alphabet A where N is the letter that follows L.

As it is difficult to express IF-ELSE statements using the SPA, we need to convert the production model to a neural model with utility functions. For example, in the case where $n = 4$, $m = 3$, we will use the set of equations shown in Table 2, where $\text{argmax}_i(U_i)$ is the action that we will take in any given state after its value has been found by the basal-ganglia-thalamus model. For this relatively simple case, we have 24 free parameters (x_i and y_i for $1 \leq i \leq 6$) which would be time-consuming to optimize before the model is initialized.

Table 1: Alphabets to be learned

Alphabet name	Contents
ALPHA	P, K, U, J, M
GAMMA	L, N, T, B, H
DELTA	V, G, I , R, F
EPSILON	W, Q, A, Z, X
ZETA	D, E, O, S, Y

The core idea here is to use the learning equation to gradually learn the connection weights between cortex and the basal ganglia, rather than using the NEF to create optimized connection weights based on the equations listed in Table 2. This avoids the problem of manually finding these param-

Table 2: Utility equations for the case $n = 4$, $m = 3$ with v as the visual system and m the working memory.

$$
\begin{aligned}
U_1 &= v \cdot x_1 \cdot \text{ALPHA} + m \cdot y_1 \cdot \text{P} \\
U_2 &= v \cdot x_2 \cdot \text{ALPHA} + m \cdot y_2 \cdot \text{K} \\
U_3 &= v \cdot x_3 \cdot \text{ALPHA} + m \cdot y_3 \cdot \text{U} \\
U_4 &= v \cdot x_4 \cdot \text{GAMMA} + m \cdot y_4 \cdot \text{L} \\
U_5 &= v \cdot x_5 \cdot \text{GAMMA} + m \cdot y_5 \cdot \text{N} \\
U_6 &= v \cdot x_6 \cdot \text{GAMMA} + m \cdot y_6 \cdot \text{T} \\
U_7 &= v \cdot x_7 \cdot \text{DELTA} + m \cdot y_7 \cdot \text{V} \\
U_8 &= v \cdot x_8 \cdot \text{DELTA} + m \cdot y_8 \cdot \text{G} \\
U_9 &= v \cdot x_9 \cdot \text{DELTA} + m \cdot y_9 \cdot \text{I} \\
U_{10} &= v \cdot x_{10} \cdot \text{EPSILON} + m \cdot y_{10} \cdot \text{W} \\
U_{11} &= v \cdot x_{11} \cdot \text{EPSILON} + m \cdot y_{11} \cdot \text{Q} \\
U_{12} &= v \cdot x_{12} \cdot \text{EPSILON} + m \cdot y_{12} \cdot \text{A}
\end{aligned}
$$

eters. Furthermore, neurons are capable of approximating much more complicated functions than the ones given in Table 2, and it may be that these more complex functions will do a better job at performing the task than the ones in Table 2, even after finding the best parameter values.

However, in order to use this learning rule, we need a supervisory training signal. That is, the learning rule requires, at each point in time, an indication as to which action should have been selected. In order to create this learning signal, we implemented the classic production-system model of the task, and used this symbolic model to create the training signal for the neural model. That is, the neural network learns-by-example from the symbolic system.

For the moment, we remain noncommittal as to where this learning signal would come from in a real biological system. Conservatively speaking, we are merely using this symbolic system as a construction tool to generate the final neuron model. That is, we are modelling the endpoint of learning, rather than the learning process itself. However, it may also be possible for a more complex neural implementation to produce a similar training signal. This is a topic for future work.

Evaluation

To assess the performance of the model, we record how many cycles it takes until the model learns each transition in every alphabet in a cycle.[1] For example, for $n = m = 2$, we find the first cycle in which the model manages to

1. complete the sequence $P \to K \to P$ or $K \to P \to K$ when the visual system contains the alphabet ALPHA
2. complete the sequence $L \to N \to L$ or $N \to L \to N$ when the visual system contains the alphabet BETA.

Our evaluation metric is therefore a single integer which captures information about how well the model responds to changes in the visual system and whether the model manages to learn each transition in each alphabet.

During our simulation, we have the learning turned on every other cycle. For the other cycles, the learning is off. As we are only interested in how the model performs when it is acting by itself rather than responding to a training signal, we only record the cycles when the learning is turned off. This means that if the model correctly learns the task once the visual system has gone through 86 cycles, we will record the number 43.

There exist other reasonable evaluation metrics, such as checking the total number of correct transitions over a time period or the total time that the working memory is reciting letters from the alphabet that is contained in the visual system. In comparison, our chosen metric does have one disadvantage; sometimes it takes a very long time for the model to learn the correct transitions, making it necessary to terminate the simulation before the model has learned the task. In that case, we report the value -1 and move on with the next simulation.

For every task, we chose to terminate the simulation when the visual system has gone through 100 cycles to limit the computational cost of running our simulations. This means that the aggregated results of the evaluation metrics are right-censored at 50 (100 divided by half since we are only care about the non-supervised cycles). This makes it non-trivial to report averages over n simulation runs or other statistical properties of the underlying distribution. To do so, we use the Kaplan-Meier estimator (Kaplan & Meier, 1958) from survival analysis in order to report the mean values.

Results

We simulated the model for 16 different versions of the task with two to five alphabets containing two to five symbols each. Our results for the learning rate 1e-10 are in Figure 2. Each data point represents the average of 30 experiments after fitting the resulting curve with an intercept-only Weibull model. We have no upper bound on the averages, but we use the 25th percentile as a lower bound. The shaded areas in Figure 2 represent the area from the lower bound to our estimate of the true average.

We collected the same results in Table 3 where the data is sorted by the total number of symbols in the model. Unsurprisingly, the number of cycles until the model learns the task increases with the number of symbols, but the number of alphabets plays a larger role than the number of letters in each alphabet; for instance, the model takes significantly longer to learn two alphabets with five letters each than five alphabets with two letters each.

We hypothesized that the dimensionality of the state representations of the letters in the alphabets could affect the results as the model might confuse two distinct letters if their neural representation is too similar. Therefore, we increased the number of dimensions from 16 to 32 and performed the same simulations. These results are shown in Figure 4. In-

[1] By a cycle, we mean the time period that it takes the visual system to display the name of each alphabet exactly once.

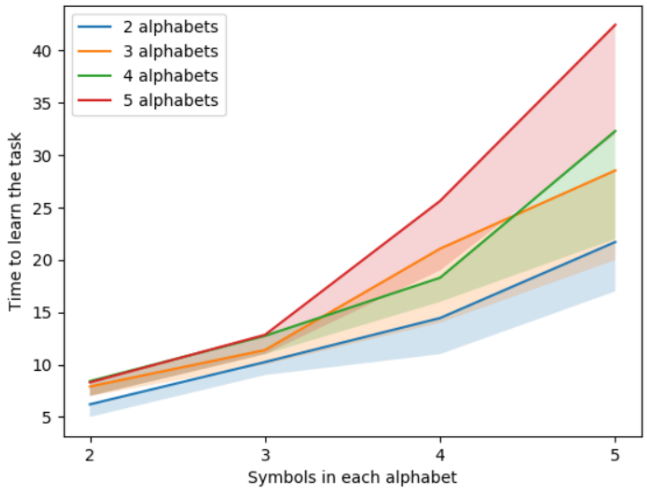


Figure 2: The evaluation metric generated with the learning rate 1e-10 and 16 dimensions for representing each symbol.

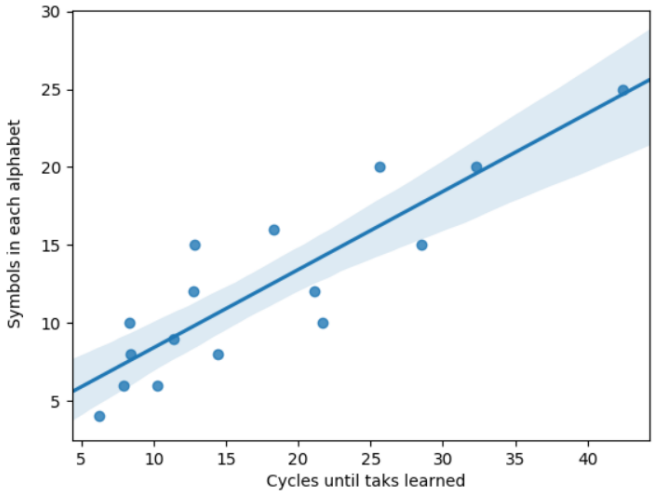


Figure 3: A scatter plot of the data from Table 3 with a linear regression model. The shaded areas represent a 95% confidence interval.

Table 3: The data from Figure 2 in tabular format, sorted by the number of symbols

Total n. of symbols	Avg. time until task learned	n, m
4	6.2	2, 2
6	7.9	3, 2
6	10.2	2, 3
8	8.4	4, 2
8	14.4	2, 4
9	11.4	3, 3
10	8.3	5, 2
10	21.7	2, 5
12	12.7	4, 3
12	21.1	3, 4
15	12.8	5, 3
15	28.5	3, 5
16	18.3	4, 4
20	25.6	5, 4
20	32.3	4, 5
25	42.5	5, 5

creasing the dimensionality resulted in much more time consuming simulations with worse results for most tasks, as can be seen by comparing Figure 2 and Figure 4.

Last, we used a different learning rate, 1e-9, to see if a higher value would lead to a faster convergence. The results of this can be seen in Table 4. The table shows that although this learning rate works well for the smaller sequencing tasks, it is much worse for the larger ones. We also tried the learning rate 1e-11, which gave worse results than shown in Table 4 for every version of the task.

Table 4: The median of the evaluation metric for 15 simulations of each learning task. Here, we used the learning rate 1e-9 and 16 dimensions.

n, m	Median of the evaluation metric
2, 2	5
2, 3	6
2, 4	10
2, 5	3
3, 2	6
3, 3	8
3, 4	22
3, 5	38
4, 2	7
4, 3	8
4, 4	24
4, 5	50+
5, 2	7
5, 3	10
5, 4	50+
5, 5	50+

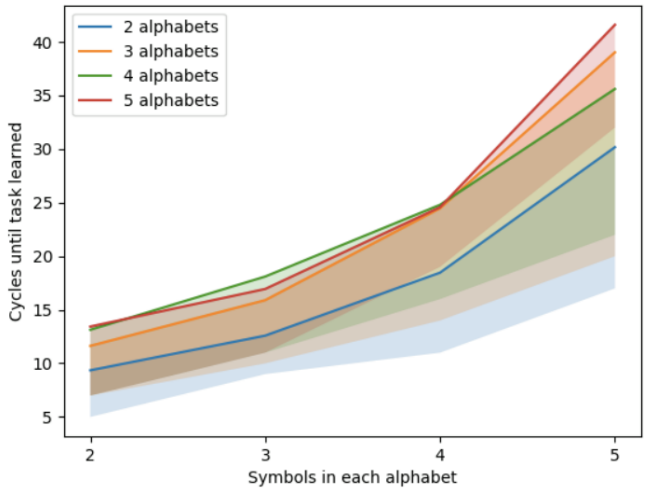


Figure 4: The evaluation metric generated with the learning rate 1e-10 and 32 dimensions for representing each symbol.

Conclusion and Further Work

The results affirm that the learning rule works well for learning utility functions for cognitive models that use neural action selection, including those constructed with the SPA. The rule appears to be scalable, as Figure 3 indicates that the time spent learning the parameters grows linearly with the scope of the task.

The results show that the value of the learning rate is important since the model fails to perform the task when the rate is too high or too low. This is a well-known phenomenon in many other learning systems, including the original cognitive model in (Stewart, Thorgeirsson, and Eliasmith, in press), which we generalized in this paper. In the future, we intend to use techniques from machine learning literature so that the system can automatically adjust the learning rate (e.g., an adaptive learning rate based on performance), thus making our model almost entirely parameter-free.

The results show that the dimensionality of the state representation of the letters used in the model also matters, but not as much as the learning rate. More investigation is needed to determine why the model is slower to learn with 32 dimensions compared to 16.

Finally, we would like to raise the question of whether the supervised learning system that we used in this paper should be considered a useful, technical tool for constructing neural models or whether it should be treated in its own right as a model of how people learn cognitive tasks. There is some evidence in favour of the latter point of view; first, the learning rule applies directly to the connections between the cortex and striatum that are used in models of reinforcement learning and second, the learning rule that we used is local and biologically plausible. In future work, we intend to explore this possibility further.

Acknowledgments

This work was supported by AFOSR grant FA9550-17-1-0026, NSERC Discovery grant 261453, and the Canada Research Chairs program.

References

Blouw, P., Eliasmith, C, and Tripp, B. (2016). A scaleable spiking neural model of action planning. *38th Annual Conference of the Cognitive Science Society*

Eliasmith, C. & Anderson, C. (2003). *Neural Engineering: Computation, representation, and dynamics in neurobiological systems*. Cambridge: MIT Press.

Eliasmith, C. (2013). *How to build a brain*. Oxford: Oxford University Press.

Gosmann, J. and Eliasmith, C. (2015). A spiking neural model of the n-back task. *37th Annual Meeting of the Cognitive Science Society*.

Kajic, I., Gosmann, J., Komer, B., Orr, R., Stewart, T.C., and Eliasmith, C. (2017). A Biologically Constrained Model of Semantic Memory Search. *Annual Meeting of the Cognitive Science Society.*

Kaplan, E. L., Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, Vol. 53, No. 282.

Kröger, B.J., Bekolay, T., and Blouw, P. (2016). Modeling motor planning in speech production using the neural engineering framework. *Electronic Speech Signal Processing (ESSV)*, 1522.

Rasmussen, D., Voelker, A.R., and Eliasmith, C. (2017). A neural model of hierarchical reinforcement learning. *PLoS ONE*, 12:7, 139.

Senft, V., Stewart, T.C., Bekolay, T., Eliasmith, C., and Krger, B.J. (2015). Reduction of dopamine in basal ganglia and its effects on syllable sequencing in speech: a computer simulation study. *Basal Ganglia* 6:1, 7-17.

Stewart, T.C., Bekolay, T., and Eliasmith, C. (2012) Learning to select actions with spiking neurons in the basal ganglia. *Frontiers in Neuroscience, 6:2, 1-14.*

Stewart, T.C., Choo, X., and Eliasmith, C. (2010). *Symbolic reasoning in spiking neurons: A model of the cortex/basal ganglia/thalamus loop*. Annual Meeting of the Cognitive Science Society.

Stewart, T.C. and Eliasmith, C. (2011) Neural cognitive modelling: A biologically constrained spiking neuron model of the Tower of Hanoi task. *33rd Annual Meeting of the Cognitive Science Society*.

Stewart, T.C. and Eliasmith, C. (2013). Parsing Sequentially Presented Commands in a Large-Scale Biologically Realistic Brain Model. *35th Meeting of the Cognitive Science Society*.

Stewart, T.C., Thorgeirsson, S., and Eliasmith, C. (in press). Supervised learning of action selection in cognitive spiking neuron models. *Proceedings of the 40th annual conference of the cognitive science society*.

The Search Space in the Eyes of the Tracker
New Indices for the cognitive load of planning

David A. Tobinski (David.Tobinski@uni-due.de)
Department of Psychology, Universitätsstr. 4
North Rhine-Westphalia, 45117-Essen Germany

Oliver Kraft (Oliver.Kraft@uni-due.de)
Department of Psychology, Universitätsstr. 4
North Rhine-Westphalia, 45117-Essen Germany

Abstract

Planning is a crucial cognitive process in many situations. The drosophila of cognitive science, the Tower of Hanoi, has been beneficial for a lot of insights. Still today it is a treasure for research around complex processes. In a quasi-experimental eye tracking study with 31 participants stable eye movement sequences have been identified in a sub-sample. Those 13 participants solved a cognitive load reducing two-discs version of the TOH. This initial study motivates for further studies and modeling methods around eye tracking data in planning scenarios.

Keywords: Tower of Hanoi, Eye Tracking, Cognitive Load of Planning

Introduction

Planning processes (PLP) are an essential part of our everyday life. We are planning on very different time-scales, like the next meal or the upcoming summer holidays, and in very different spaces, for example within our current surrounding or in very far vacation destinies. If planning is not available, a problem is on the horizon. And even the solution of the problem dependents deeply on PLP. In our perspective PLP takes place in working memory (WM; Baddeley, 2000). The information within the planning task leads to a potential cognitive load of planning (CLP). CLP can be reduced by a subgoaling strategy, which is described by Simon (1975) in the Tower of Hanoi (TOH) research paradigm. Gunzelmann and Anderson (2003) describe different planning and learning strategies within isomorphs of the Tower of Hanoi (TOH), where learning strategies are identified as decoding processes between WM and long-term memory (LTM). Our question is whether the process of subgoaling is only represented in internal states between WM and LTM or is it possible to identify subgoaling on the surface? Therefor an eye tracking study has been conducted.

The Tower of Hanoi

The Tower of Hanoi (TOH) can be seen as the drosophila of cognitive science. The task consists of three stacks (A,B,C) and a defined amount of disks of different size. Mostly three or four disks are used in experiments. These discs have to be moved from an initial stack (A) to a specific goal stack (C) under three rules: (1) only one disk can be moved at a time, (2) a larger disks is not allowed to be placed above a smaller disc (called the golden rule) and (3) the fastest path has to be identified. If three disks have to be moved a minimum of seven moves are necessary for the optimal solution path regarding all conditions. Because of the golden rule the path is completely determined. Thus all moves depend on each other and even the first decision to place the first disc on stack B or C depends on the complete plan. Regarding this fact all moves have to be planned before the first disc is placed on stack B or C. In case of three discs the complete search space consists of 27 possible states under the condition of the golden rule. Disregarding this rule leads to 24 additional forbidden states and a possible search state of 51 states at all.

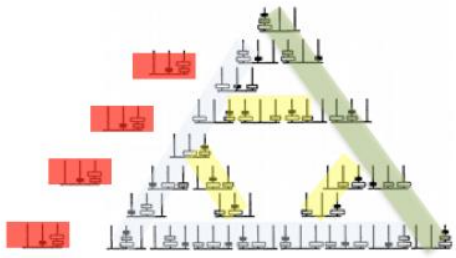


Figure 1: The 27 allowed search space states of TOH-D3 are shown within the traditional trajectory. The red marked states are only possible, if the golden rule is disregarded.

Objectives

While reaction times in the TOH have been very well analyzed within the last decades there exists a lack of eye tracking data. (RQ1) Can eye tracking data be used for an interpretation of search space trajectories?
(RQ2) Might eye tracking data be a predictor for different planning strategies, which are crucial in managing the CLP?

Method

Participants

A total of 31 participants from the University of Duisburg-Essen have been recruited for a “Tower-of-Hanoi-Eye-Tracking-Study” with 2 disks, 3 disks and 4 disks Versions of TOH. The presented sub-sample of the study consists of 13 educational master-students (nine female; four male; mean of age: 25.15, sd: 6.45) from the University of Duisburg-Essen, North Rhine-Westphalia, who worked with the 2 disk and 3 disk Version.

Materials

A digital version of the Tower of Hanoi (TOH-D) was implemented with JavaScript and the logfiles are coded in JSON. For better insights in eye movement sequences we were interested in a very reduced CLP, therefor we used initially a two disc version (TOH-D2).
The THO-D was presented on a screen-based eye tracker (SMI RED500; 500 Hz; 0.4 degrees, 22" display). Three areas of interest (AOIs) have been edited around the three stacks. The raw data of the SMI RED500 was exported to csv-files and analyzed with R (R version 3.4.3) and the TraMineR package.

Procedure

All participants have been introduced to the ethical principles of the University of Duisburg-Essen, which are compatible to the ethical principles of the German Psychological Society (DGPs). The quasi-experimental study took place in a highly standardized Eye Tracking Laboratory at the Institute of Psychology. TOH-D starts with a task independent mouse-movement training. Afterwards all participants have to solve TOH-D2 and the three-disc version (TOH-D3).

Results and Discussion

In TOH-D2 two participants have been excluded because of missing data. From the resulting 11 participants ten reached the goal with the needed three moves. One participant reached the goal with four moves and additionally disregarded the golden rule. The mean of the solution time is *12198.31 milliseconds* with a standard deviation of *5398.4 ms*. Sequences have been produced in an alphabet (A-B-C) of AOI fixations. The minimum of an AOI fixation time was set to 50ms (Anderson, 2007).

RQ1

Concerning RQ1 very stable eye movement sequences can be identified in the first planning phase (B-A-B) in the two-discs TOH. The weak performing participant shows the most complex search pattern in the first planning phase (B-A-B-A-B-C-B-C-B-C-B).

RQ2

On the search for different strategies transition matrices between AOIs can be regarded. There are strong differences in the eye movement behavior between high and low TOH performer. The low performer show more movement on stack C. Stack C concerns the goal state. Our interpretation is that eye-movement might help to reduce CLP, the information is more processed on the surface of its source. Thus the better performer do not need to represent the goal state on the surface level. What is the case in both performing groups for the sub-goals (Stack B).

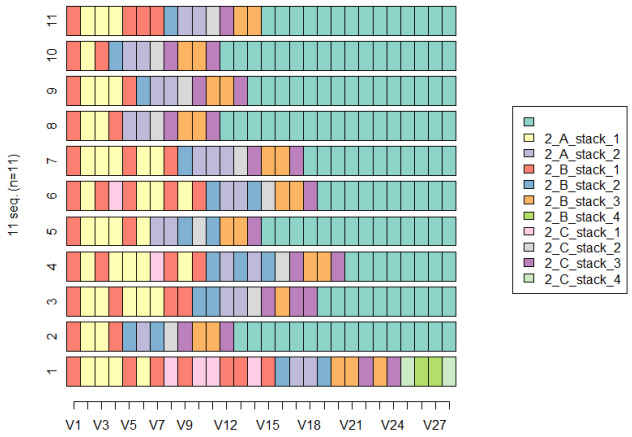


Figure 2: The search space and eye movement sequences between three areas of interest (AOIs: _A_,_B_,_C_) in the TOH-D2. Only one person (p1) needed four moves and planning phases (_4). Stable search sequences can be identified between the participants.

Conclusion

Eye tracking data seems to be promising for interpreting PLP in the TOH research paradigm. The current AOIs only represent the eye movement on and between the stacks. In a special TOH-D version for eye tracking studies the discs might be bigger for defining AOIs on the level of the discs. This will lead to a more precise differentiation of the PLPs. The most important step will be the work on the model of the different buffer and the perceptual-motor modules of the ACT-R architecture.

Table 1: The transition matrices of the first planning phase of the three-discs TOH (TOH-D3) show differences between high (n=6) and low performer (n=6) especially on stack C.

	-> A	**-> B**	**-> C**
A ->	.44/.47	.39/.48	.00/.03
B ->	.59/.39	.24/**.41**	.00/**.17**
C ->	.00/**.21**	.00/**.63**	.00/**.11**

References

Anderson, J. R. (2007). How Can the Human Mind Occur in the Physical Universe? Oxford: Oxford University Press.

Gunzelmann, G., & Anderson, J. R. (2003). Problem solving: Increased planning with practice. Cognitive Systems Research, 4(1), 57–76.

Baddeley, A. D. (2000). The episodic buffer: a new component of working memory? Trends in Cognitive Sciences, 4(11), 417–423.

Gabadinho, A., G. Ritschard, M. Studer & N. S. Müller (2010). Mining sequence data in R with the TraMineR package: A user's guide, University of Geneva.

Simon, H. A. (1975). The functional equivalence of problem solving skills. Cognitive Psychology, 7, 268–288.

Automatically translating logical strategy formulas into cognitive models

Jakob Dirk Top (scholar@jakobdirktop.nl)
Institute of Artificial Intelligence, Nijenborgh 9,
Groningen, 9747 AG, the Netherlands

Rineke Verbrugge (l.c.verbrugge@rug.nl)
Institute of Artificial Intelligence, Nijenborgh 9,
Groningen, 9747 AG, the Netherlands

Sujata Ghosh (sujata@isichennai.res.in)
Indian Statistical Institute, 110 Nelson Manickam Road,
Aminjikarai, Chennai 600029, India

Abstract

Whereas game theorists and logicians use formal methods to investigate strategic behaviour, cognitive scientists use cognitive models of the human mind to predict and simulate human behaviour. In this paper, we hope to bring these fields together by creating a translation system which, starting from a strategy represented in formal logic, automatically generates a computational model in the PRIMs cognitive architecture. We run such models to generate response times and decisions made in centipede-like games, a subset of dynamic perfect-information games. Our system is a proof-of-concept for generating cognitive models from formal logic, and presents a new method of otherwise laborious model creation.

Keywords: formal logics, PRIMs, strategic reasoning, automated model generation

Introduction

Centipede-like games

In this paper, we model participants' reasoning in a turn-taking game called Marble Drop (Figure 1). Participants played this game in an experiment against a computer opponent (Ghosh, Heifetz, Verbrugge, & De Weerd, 2017). Because the structure of the game is reminiscent of a centipede (its body extends from top left to bottom right of Figure 1 along the trapdoors and it has five feet corresponding to the bins containing the marbles that are the players' payoffs), such games are dubbed 'centipede-like games'.[1]

Game theory prescribes that players who are commonly known to be rational use the *backward induction* (BI) strategy: one should ignore previous information, and work backwards from the end of the game tree to reach a decision (Perea, 2012). For example, in the 'orange' player's last turn in Game 1 (Fig. 1), he has to decide between going to the left or to the right, for payoffs of 4 or 3 orange marbles, respectively. Using BI, because 4 is more than 3, he chooses to go left, delivering the outcome pair (1,4): 1 for the blue player, 4 for the orange player. One can then continue backwards to compare the left and right choices in the blue player's second turn: going right gives (1,4) while going left gives (3,1); because 3 is more than 1, the blue player would choose to open the left blue trapdoor. One then continues to reason backwards to compare the actions in the orange player's first turn, where the outcome is (1,2) when playing left and (3,1) by playing right. One assumes that, 2 being more than 1, the orange player chooses to open the left orange trapdoor. Finally, one compares the actions in the blue player's first turn, where going left leads to (4,1) and going right leads to (1, 2). Because 4 is more than 1, the blue player will choose to open the left trapdoor to obtain 4 points. Note that playing rationally by backward induction does not necessarily lead to the outcome with the highest sum of players' payoffs – that would have been achieved by both players choosing to open their right trapdoors at all four decision points and ending up with a combined payoff of 6+3.

[1]The games in this paper do not comply with the conditions on payoffs of Rosenthal's original centipede game (Rosenthal, 1981).

Cognitive models

We can investigate human behaviour in centipede-like games by constructing computational cognitive models of the mind and comparing their behaviour to human behaviour. We assume full understanding of the game's rules for both the human players and cognitive models, and investigate their gameplay and the strategies they may be using. We create these models in the PRIMs cognitive architecture (Taatgen, 2013). We recall a few key aspects of PRIMs models. Models in PRIMs have a working memory, used as a mental scratch-pad, and a declarative memory, used for long-term storage of information. Visual information is presented to a PRIMs model hierarchically; the model uses focus actions to move its visual attention through layers of the hierarchy. Models in PRIMs operate by sequentially firing *primitive elements*, which move or compare information present in the model or in the visual input it receives. The process of *production compilation* compiles primitive elements that are often fired in the same sequence into a single production, causing a speed-up when performing the same task multiple times.

Formal logic

In (Ghosh & Verbrugge, online first), a formal logic is proposed which formalizes strategic behaviour as demonstrated by human participants in centipede-like games. A formula describing a strategy, a *strategy formula*, consists of a set of

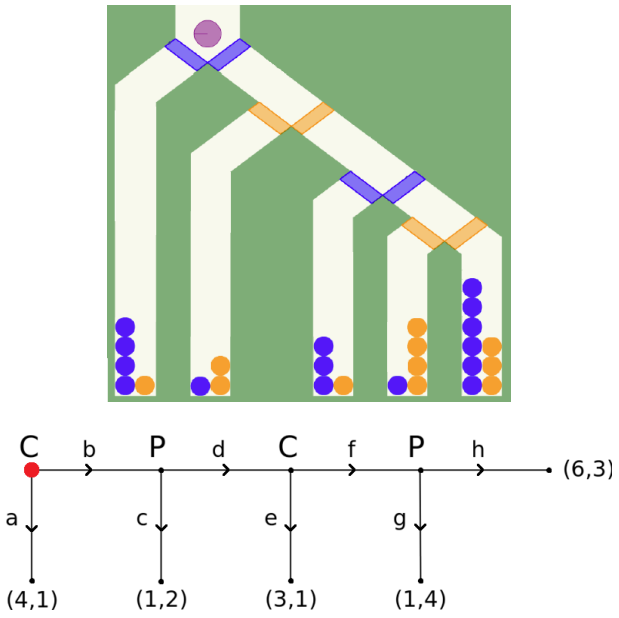


Figure 1: Top: Marble Drop game. Players (assigned blue and orange) control the marble's course by opening the left or right trapdoor of their color once the purple marble arrives there. When it ends up in a certain bin, each player earns the marbles of their color. This example payoff structure corresponds to Game 1 of (Ghosh et al., 2017), see bottom figure. In the payoff pairs, the left payoff is C's and the right is P's.

conditions and an action. If the conditions hold, the action should be played. We use a *myopic strategy* as an example: a player using the myopic strategy only looks at his own payoffs at the current and next ending location. Consider a play of Game 1 starting at player C's first turn. Using the myopic strategy, player C looks at his own payoff should he play a, which is 4, and compares it to his own payoff should he play b and should player P play c, which is 1. The former is larger so player C should play a using the myopic strategy. This case is captured by strategy formula $\mathcal{K}_C^1$ as follows:
$[(\langle a^+\rangle(u_C = 4) \wedge \langle b^+\rangle\langle c^+\rangle(u_C = 1) \wedge (1 \leqslant 4) \wedge \textbf{root}) \mapsto a]^C$

Here, $\mathcal{K}_C^1$ is the name of the formula, with the game number in superscript and the player in subscript. The formula itself is followed by its corresponding player in superscript. The formula consists of conditions, separated by conjunction symbols ($\wedge$), as well as an action, in this case a. The conditions and action are separated by a mapping arrow ($\mapsto$). The first condition, $\langle a^+\rangle(u_C = 4)$, specifies that after edge a is traversed ($\langle a^+\rangle$) from the currently active node (in this case the first node, marked red in Figure 1), player C's payoff should be 4 ($u_C = 4$). The second condition, $\langle b^+\rangle\langle c^+\rangle(u_C = 1)$, specifies that after edges b and c have been traversed, player C's payoff should be 1. The third condition, $(1 \leqslant 4)$, indicates a comparison between these two payoffs. The last condition, **root**, specifies that the currently active node should be the root of the game tree, which ensures that moving across an edge using an operator such as $\langle a^+\rangle$ is possible at this location. If all of these conditions hold, then player C plays a.

Research goals

In this paper we propose a system which creates a PRIMs model from a strategy in the formal logic we just described, capable of playing centipede-like games. Our encompassing goal is to help understand human behaviour in dynamic perfect-information games. Our place in this continuing body of research can be found in Figure 2. Here, human behaviour is found at the top of the diagram, as all research involved aims to understand human behaviour. By observing human behaviour, game theorists formalize strategies as possibly used by human participants. These formal strategies can be used by cognitive scientists to manually construct cognitive models. These models automatically generate data, such as response times. The blue arrow signifies the classic approach of creating cognitive models by hand based on observed human behaviour. The behaviour of such a model can be verified by constructing a behavioural experiment, which gives us data about human behaviour, closing the circle. In the diagram, dashed lines are automated processes. The red dashed line indicates our current research, which automates the creation of cognitive models from formal strategies.

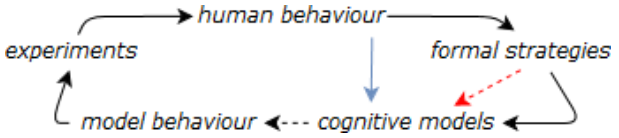


Figure 2: The red arrow in this diagram indicates our place in research, as we aim to automate the creation of cognitive models from formal strategies.

The primary goal of this research is to create a system which automatically generates a fully functioning model in the PRIMs cognitive architecture, capable of playing centipede-like games, from a strategy represented in the formal logic from (Ghosh & Verbrugge, online first). To achieve this goal, we solve several subgoals: first, we create a model in the PRIMs cognitive architecture, capable of playing these games, by hand. We need this model to verify our translation system. Next, we discuss the differences between the formal logic and cognitive models, which is essential to translate one of them into the other. In the next section we describe our solutions and our translation system.

Methods

Verification model

First, we create a cognitive model in the PRIMs cognitive architecture by hand, which uses the myopic strategy as discussed in the introduction. This model plays as player P. We refer to it as the *myopic model*. We only look at the model's first move, in games that either start with player P, or in games where player C has already played action b. We use Game

1 in Figure 1 to demonstrate how this model operates. A model's visual attention always starts at the root of the tree, which is the red dot in the image. The myopic model first moves its visual attention to the first ending location that may be reached. In Game 1, it moves its visual attention across edges b and c. The model then stores its own payoff at this location, the value 2, in its first slot of working memory. The model proceeds to move its visual attention across edges c, d, and e to look at the next ending location. Now the model stores its own payoff at this second location, the value 1, in its second slot of working memory. Lastly, the model retrieves a chunk from declarative memory to compare the values 2 and 1 which it has stored in working memory. The model succeeds in retrieving a chunk specifying that 2 is larger than 1 and, based on this information, ends the game by playing c. This myopic model is generic - it can use the myopic strategy in any centipede-like game. According to the own-payoff strategy, a player should only look at their own payoffs in a game tree, and try to move towards that payoff. For example, in Game 1 in Figure 1, player C should play b and f in an attempt to reach the payoff of 6 at the rightmost node, which is the game's highest payoff. The *own-payoff model* behaves in a manner similar to the myopic model. However, it will also move its visual attention to and make comparisons between its own payoffs further along the game tree. If it discovers a payoff higher than its first payoff, it will move right. If it has compared all payoffs to the first one, and the first one is the largest payoff, it will move down.

Translation system

As mentioned, our new translation system automatically generates a model in the PRIMs cognitive architecture from a formal strategy and its corresponding game. To do so, we first need to represent centipede-like games and formal strategies in our system. For the games, we use the same tree-like structure as seen in Figure 1. Centipede-like games consist of nodes and edges. Nodes can be leaf nodes (ending locations) and non-leaf nodes (player turns). For leaf nodes, both players' payoffs are specified. For non-leaf nodes, the player who has a turn at the node is specified. Edges specify the two nodes they connect. All finite centipede-like games can be stored in this manner. Formal strategies consist of a player, an action, and a list of conditions. Each of these conditions consists of a list of zero or more operators (such as $\langle a^+ \rangle$), and a proposition (such as $(u_C = 4)$). We have already seen most of these propositions and operators in the formula $\mathcal{K}_C^1$.

We now give truth definitions of the relevant logic formulas used in our translation system. The truth of a formula ψ at a node s is defined inductively as follows (see also Ghosh and Verbrugge (online first)). Here, $M = (T, \{\longrightarrow_i^x\}, V)$ is a model consisting of a game tree T, a binary relation on the nodes of the tree corresponding to each node x and each player i (denoted by $\{\longrightarrow_i^x\}$), and a map V assigning to a state s the set $V(s)$ of all true propositions in s.

1. $M, s \models p$ iff $p \in V(s)$ for atomic formulas p.
2. $M, s \models \langle a^+ \rangle \psi$ iff there exists an s' such that $s \stackrel{a}{\Rightarrow} s'$ and $M, s' \models \psi$.
3. $M, s \models \langle a^- \rangle \psi$ iff there exists an s' such that $s' \stackrel{a}{\Rightarrow} s$ and $M, s' \models \psi$.
4. $M, s \models \mathbb{B}_{\mathbb{h}}^{(i,x)} \psi$ iff the underlying game tree T_M is the same as the one for $\mathbb{h}$ and for all s' such that $s \longrightarrow_i^x s', M, s' \models \psi$.

In layman's terms: formulas in the logic are interpreted at the currently active decision node, or the current turn, except when they are preceded by an operator $\langle a^+ \rangle$. Such an operator indicates that the remainder of the statement should be interpreted at the location obtained by following edge a. The proposition **root** is true if the node it refers to is the root of the tree, or the first turn of the game. The proposition $(u_i = q_i)$ is true if player i's payoff is equal to q_i at the node it refers to. The proposition $(r \leqslant q)$ is not interpreted at a specific position, and is true if r is equal to or smaller than q. The proposition $\mathbf{turn}_i$, not found in the formula $\mathcal{K}_C^1$, is true if it is player i's turn at the node it refers to. Finally, we have formulas such as $\mathbb{B}_{g1}^{n1,C} \langle b^+ \rangle c$. This one means 'player C, in Game 1, at the first node, believes that after playing b, c will be played.' It consists of a belief operator $\mathbb{B}_{g1}^{n1,C}$, which accounts for the phrase 'player C, in Game 1, at the first node, believes that'. The operator $\langle b^+ \rangle$ accounts for 'after playing b', and c accounts for 'c will be played'.

Individual components In the present paragraph, we give for each novel component in the logic of Ghosh and Verbrugge (online first) the corresponding behaviour of a PRIMs model generated using this component.

$\langle a^+ \rangle$ **and** $\langle a^- \rangle$: Operators such as $\langle a^+ \rangle$ and $\langle a^- \rangle$ indicate that a proposition should be evaluated at a location different from the current location, and specify which location. A model translated from a formula containing these operators uses focus actions to move its visual attention to the specified location. Focus actions take time to complete similar to human gazing, causing these operators to increase the model's reaction time.

root: When a strategy formula contains the proposition **root**, the PRIMs model will visually inspect the specified node to determine whether it is the root of the tree.

$\mathbf{turn}_i$: When a strategy formula contains a proposition $\mathbf{turn}_i$, where i is C or P, the PRIMs model will read the player name from the specified node in the game tree, and compare it to i.

$(\mathbf{u_i} = \mathbf{q_i})$: The proposition $(u_i = q_i)$ states that player i's payoff is equal to q_i at a certain location. The PRIMs model will compare q_i to a value in its visual input. Because this value may be required for future comparisons, it is also stored in an empty slot of working memory.

$(\mathbf{r} \leqslant \mathbf{q})$: A proposition $(r \leqslant q)$ is a comparison between two values in the game tree. A PRIMs model cannot instantly access each value in a visual display: it has to remember them by placing them in working or declarative memory before it can compare them. A proposition $(u_i = q_i)$ causes

such a value to be stored in working memory. A proposition $(r \leqslant q)$ then sends two of these values from working memory to declarative memory, to try and remember which one is bigger. When a model is created, its declarative memory is filled with facts about single-digit comparisons, such as $(0 \leqslant 3)$ and $(2 \leqslant 2)$.

$\mathbb{B}_{\mathbb{h}}^{(i,x)}$ **and** a: The operator $\mathbb{B}_{\mathbb{h}}^{(i,x)}$ and proposition a are used to construct beliefs about the opponent's strategy. In a belief operator, i is one of the players C or P. The symbol x is the decision node, or turn, where the belief is held. For the four turns in Game 1 (Figure 1) we use *n1*, *n2*, *n3* and *n4*, respectively. Lastly, $\mathbb{h}$ is the game the belief applies to. In Game 1, $\mathbb{h}$ is *g1*. The symbols a through h refer to the actions that can be played in the games, represented as propositions. An example belief is expressed in the following formula:

$$\mathbb{B}_{g1}^{(C,n1)}\langle b^+\rangle\langle d^+\rangle e \tag{1}$$

This formula means 'In Game 1, at node 1, player C believes that after playing of b and d, e will be played'. To verify such a belief, a model employs a strategy similar to the ones used by models in (Stevens et al., 2018). When a model is created, it contains several player-specific strategies in its declarative memory. When a model verifies a belief, it sends a *partial* sequence of actions to declarative memory, corresponding to the assumptions of the belief, in an attempt to retrieve a *full* sequence of actions, which is a strategy. Using Equation 1 as an example, the assumptions of the belief are that b is played. Therefore the model sends the sequence b to declarative memory. The sequences *b-e* and *b-f* could be retrieved, depending on the strategies present in declarative memory. However, only *b-e* verifies $B_{g1}^{(C,n1)}\langle b^+\rangle\langle d^+\rangle e$. All others falsify it.

Problems We encountered two problems in the formal logic in the previous section. First, a comparison $(r \leqslant q)$ does not specify *which* payoffs are being compared, only which two natural numbers (including zero) are being compared. A game containing identical payoffs at different nodes poses a problem for a translation system. Although humans can intuitively determine which comparison would 'make sense', a translation system cannot. Because of this, we use a modified version of the formal logic where each payoff is marked, and each comparison refers to two specific payoffs. This allows a translation system to know precisely which two payoffs it has to pull from working memory to perform a comparison.

Secondly, a belief such as $\mathbb{B}_{g1}^{(C,n1)}\langle b^+\rangle c$ only specifies *what* should be believed, not *how* this belief should be obtained. We use a method similar to (Stevens et al., 2018), where the model begins with strategy chunks in its declarative memory, and verifies a belief by comparing the current game state to these strategy chunks. We have strategy chunks for three strategies: (i) the extensive-form rationalizable (EFR) strategy, which we use because the actions it prescribes correspond more closely than backward induction to the human data of Ghosh, Heifetz, and Verbrugge (2015), (ii) the backward induction (BI) strategy, which we use because it reaches a Nash equilibrium and is often put forward as the game-theoretical solution to games similar to our own (see explanation in Introduction), and (iii) the own-payoff strategy, which is used in previous research on centipede-like games, such as (Ghosh et al., 2017).

According to the EFR strategy, one should consider previous information (Perea, 2012). For example, if player C plays b in Game 1, then player P could rationalize this decision by believing that C has skipped the 4 points obtained by playing a, because C believes that he can get the 6 points at the far right (the only payoff higher than 4). To get these 6 points, player C has to play f as well, which P can use in his decision to play c or d.

Let us explain the model behaviour for a complete strategy formula, namely $\mathcal{K}_C^1$. This formula, as well as the formulas used in previous research using this particular logic, takes the form of a Horn clause, such as $a \wedge b \wedge c \wedge d \rightarrow p$. Given a strategy formula, a PRIMs model tries to verify each proposition in the conjunction sequentially, using the behaviour earlier described for each proposition. If it encounters a proposition it cannot verify, it 'jumps out of' this verification process and *doesn't* play the action prescribed by the formula (what it does we describe in the next section). If the model has verified all the propositions in the conjunction, it plays the action prescribed by the strategy formula.

One problem remains. Conjunctions in formal logic are unordered. Models in PRIMs, however, solve problems sequentially. Therefore we need to order the conjunctions in formal logic, so the corresponding PRIMs model has an order to verify them in. Fortunately, each proposition has to be verified at a specific location. For example, $\langle a^+\rangle(u_C = 4)$, in Game 1 found in Figure 1, has to be verified at the ending location reached by playing a. Eye-tracking data from Meijering, Van Rijn, Taatgen, and Verbrugge (2012) tells us that human participants tend to look through a game tree by following the edges along the shortest path. Therefore we compute the shortest path through the game tree. Our PRIMs models verify propositions as they occur along this path.

Exhaustive strategy formulas In the previous sections we described how a PRIMs model, generated by our translation system, behaves based on a strategy formula. One strategy formula is not always sufficient to describe a strategy. Strategies such as BI and EFR can have multiple solutions if there are payoff ties: in Game 4 in Figure 3, the last two payoffs for P, obtained by playing g and h, are tied, allowing for two options when performing the BI procedure. When using EFR in Game 4, player C playing b instead of a can be interpreted as C going for any of two payoffs higher than the 2 he skipped by playing b. Thus, one has to list all solutions for the specified strategy. Therefore, our translation system allows for a strategy to consist of a list of strategy formulas. The PRIMs model generated from this list tries to verify each formula in it, using the behaviour described in the previous sections, until it finds one it can verify, and play the action prescribed by the formula it verified. There is no need to specify what the

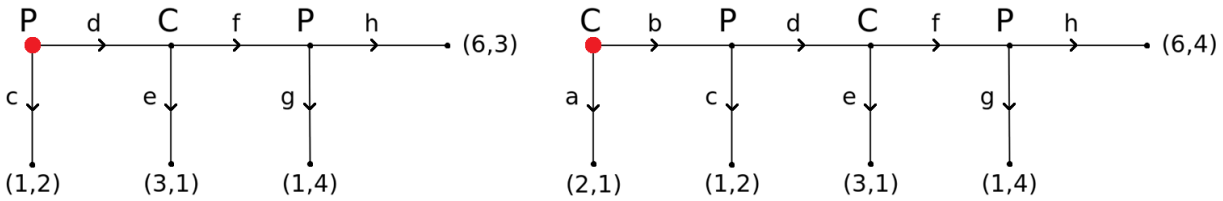


Figure 3: Games $1'$ and 4 from (Ghosh et al., 2017). Game $1'$ is on the left, Game 4 is on the right.

model has to do when it cannot verify any of the formulas: the list is exhaustive, so at least one of them holds.[2]

Experiments

Verification experiment In our verification experiment we compare our handmade with our automatically generated models, based on six different games, three of which can be found in Figure 1 and Figure 3. We have four models: handmade myopic and own-payoff models, and automatically generated myopic and own-payoff models, which are generated from the myopic and own-payoff strategy formulas for Game $1'$. The formula for the myopic strategy is as follows:

$My_P^{1'} : [(\langle c^+ \rangle (u_P = 2) \wedge \langle d^+ \rangle \langle e^+ \rangle (u_P = 1) \wedge (1 \leqslant 2) \wedge \mathbf{root}) \mapsto c]^P$

These models only play Game $1'$ (Figure 3), in the role of player P against computer opponents C playing prespecified moves. We use the same methods as the models in (Ghosh & Verbrugge, online first): we run each model 50 times, where it plays 50 games, to simulate 50 virtual participants who play 50 games each. We record reaction times and decisions.

Differences In our handmade models we have implemented the procedural knowledge required to play the myopic and own-payoff strategies, as described at the beginning of the 'Methods' section. Our automatically generated models, in contrast, play by sequentially verifying propositions in a list and play a certain action if all of them hold, according to a logical formula. Our handmade models are general for any centipede-like game, whereas our automatically generated models are specific for one game. Due to their generality, our handmade models have to look at more properties of the game tree, because no assumptions can be made. For example, for the myopic strategy, when reading the first payoff value from the game tree, for the handmade model, its current goal is 'finding a value in the game tree', its visual attention has to be directed at a leaf node, two slots of working memory have to be empty, one to store the first value, and another has to remain empty for a next value. For the automatically generated model, the game is assumed to be known, as well as the sequence of actions the model has to fire. Therefore, when reading the first payoff value from the game tree, for the automatically generated model, its current goal is 'finding the first value in the game tree', its working memory states that this is the second action performed in this goal, and within its visual input the payoff value the model is currently looking at should be 'two' (in order to verify $\langle c^+ \rangle (u_P = 2)$).

In short, our handmade models are implementations of a strategy, whereas our automatically generated models perform a sequence of actions corresponding to a logical formula, based on this strategy. We compare them to verify our translation system and to investigate the differences.[3]

Exploratory experiment In our second experiment we explore the abilities of our translation system by looking at novel automatically generated models. These models play two games: Game 1 (see Figure 1), and Game 4 (see Figure 3). For both games, we automatically generate two models: one that uses backward induction, and one that uses extensive-form rationalizability. These four models are generated from the BI and EFR strategy formulas for Game 1 and Game 4. These strategy formulas not only contain payoffs and comparisons, like the myopic and own-payoff strategy formulas, but also contain beliefs. We use these games because both the BI and EFR solutions differ between these games, which should give different results. Because some of these strategies have multiple solutions in some of these games, exhaustive strategy formulas are required to describe these strategies, whereas we did not use exhaustive strategy formulas in the previous experiment.

Results

Verification experiment In terms of behaviour, myopic models always play down, and own-payoff models always play right. This corresponds to the strategies these models were generated from. The reaction times for our four models can be found in Figure 4. It indicates that the myopic models are faster than the own-payoff models, and the generated models are faster than the handmade models. Comparing the handmade and automatically generated models, the proportional difference in mean reaction times between the myopic and own-payoff models is similar. The proportion myopic/own-payoff is 0.56 for the handmade models, and 0.55 for the generated models.

[2] The strategy specification language described in (Ghosh & Verbrugge, online first) provides two possible ways of combining strategy specifications, namely, $n_1 + n_2$ (n_1 or n_2) and $n_1 \cdot n_2$ (n_1 and n_2). While the former corresponds to the exhaustive list of strategies, the latter operator is yet to be modelled.

[3] We did not create, by hand, myopic and own-payoff models that play by verifying a strategy formula, for two reasons: firstly, our translation system never fails at generating a cognitive model, and secondly, given a strategy formula, the behaviour of a generated model is fully known to us - writing the same model by hand would be akin to 'copying' the output of our translation system.

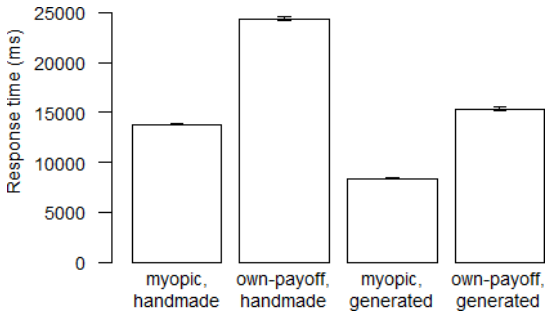


Figure 4: Reaction times of handmade and system-generated myopic and own-payoff models at first decision in Game $1'$.

Exploratory experiment The reaction times for our BI and EFR models can be found in Figure 5. Our exhaustive mod-

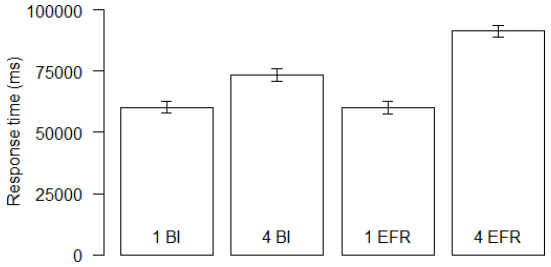


Figure 5: Reaction times for our automatically generated BI and EFR models, when making their first decision. Here, '1 BI' is the reaction time for the BI model in Game 1, etc.

els are a lot slower than our myopic and own-payoff models. Furthermore, reaction times in Game 1 are faster than reaction times in Game 4. It seems that reaction times are a function of the number of formulas required to create the exhaustive strategy formula: for both BI and EFR, in Game 1, only one formula is needed. In Game 4, BI requires two formulas, and EFR requires four formulas. To test this, we perform a simple linear regression using number of formulas to predict reaction times. A significant regression equation is found ($F(1,189) = 432.6, p < 2.2 \cdot 10^{-16}$), with an R^2 of 0.696. Predicted reaction time in milliseconds is equal to $10401 + 50453 \cdot$(number of formulas).

Discussion

The results in the previous sections show the feasibility of our system as a proof-of-concept. For all turn-taking games with at most binary choices, our system generates cognitive models from strategy formulas, without human intervention. This can greatly speed up research, because cognitive models are often created by hand. These models can be run to obtain reaction times, as shown in our results, as well as other data, such as decisions, gazing behaviour, and neural activity. Our verification experiment shows that between the handmade and generated models, the proportion of reaction times between the myopic and own-payoff strategies are highly similar, and the actions they play are the same. We believe this is due to the similarity in their decision-making processes. For example: in both myopic models the model looks at two payoffs in the game tree, stores them in memory, and compares them to make its decision. The difference in reaction times could be due to the following reason: the automatically generated models are specific models for their respective games, whereas the handmade models are general models. Therefore, the handmade models have to perform extra tests to verify whether certain operations are possible in the current game, and to remember what they have already done. These extra steps cannot be removed: the model cannot function without them. However, the generated models are generated from a strategy formula designed for a particular game, so they can simply perform a sequence of actions. This corresponds to the number of primitive elements required by these models. For example, the automatically generated myopic model uses 39 primitive elements, and the handmade myopic model uses 46 primitive elements.

Nonetheless, our exploratory experiments show that our system can provide predictions such as 'human response times in centipede-like games are a function of the number of game-theoretical solutions in a game', which can be experimentally tested. In the future we plan to extend the automatic translation method to perfect-information games with more-than-binary decision points. Similar translation methods of linking formal logic directly to cognitive modeling and behavioural experiments may be constructed for wider classes of tasks, such as planning and communication protocols.

References

Ghosh, S., Heifetz, A., & Verbrugge, R. (2015). Do players reason by forward induction in dynamic perfect information games? In R. Ramanujam (Ed.), *Proc. 15th conf. theor. aspects rationality and knowledge* (pp. 121–130).

Ghosh, S., Heifetz, A., Verbrugge, R., & De Weerd, H. (2017). What drives people's choices in turn-taking games, if not game-theoretic rationality? In J. Lang (Ed.), *Proc. 16th conf. theor. aspects rationality and knowledge* (pp. 265–284).

Ghosh, S., & Verbrugge, R. (online first). Studying strategies and types of players: Experiments, logics and cognitive models. *Synthese*, *2018*.

Meijering, B., Van Rijn, H., Taatgen, N. A., & Verbrugge, R. (2012). What eye movements can tell about theory of mind in a strategic game. *PLoS ONE*, *7*(9), e45961.

Perea, A. (2012). *Epistemic game theory*. Cambridge UP.

Rosenthal, R. (1981). Games of perfect information, predatory pricing and the chain-store paradox. *Journal of Economic Theory*, *25*(1), 92–100.

Stevens, C. A., Daamen, J., Gaudrain, E., Renkema, T., Top, J. D., Cnossen, F., et al. (2018). Using cognitive agents to train negotiation skills. *Frontiers in Psychology*, *9*.

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471.

Modeling the Impact of Fake News on Citizens

Stephanie Tulk (stulk@GMU.Edu)
Department of Psychology, George Mason University
Fairfax, VA 22030 USA

Niloofar Bagheri-Jebelli (nbagher2@GMU.Edu)
Computational Social Science Program, George Mason University
Fairfax, VA 22030 USA

William G. Kennedy (wkennedy@GMU.Edu)
Center for Social Complexity, George Mason University
Fairfax, VA 22030 USA

Abstract

The impact of "fake news" on the 2016 presidential election became a serious concern after the surprising results. The volume of fake news on social media, which people used as a serious news source, could have significantly affected voters' opinions. It is important to consider how social and cognitive processes were affected by this fake news to estimate the true impact of this computational propaganda technique. We built a cognitive model of a citizen deciding what to believe when encountering election stories on social media, eventually developing an opinion and using motivated reasoning to help determine which stories are true. Modeling 100 citizens, we assemble polls of the agents over the 9 months leading up to the election that replicates the qualitative characteristics of actual polls but leaves many questions outside the purview of cognitive modeling.

Keywords: Cognitive modeling; opinion modeling; motivated reasoning; computational social science.

Introduction

After the 2016 presidential election in the United States, "fake news" became a topic of concern due to a surge of false news articles viewed and shared on social media in the months leading up to the election. Further investigation revealed that engagement with fake news articles on Facebook outnumbered those of real news (Silverman 2016) and the shared fake news was much more likely to be pro-Trump (the Republican candidate) than pro-Clinton (the Democrat) (Allcott & Gentzkow, 2017). It is worth considering that rather than being insignificant noise, the biased fake news may have influenced voters enough to impact the election results. This new misinformation approach to influencing public opinion and its effectiveness make up an interesting cognitive phenomenon.

We investigated the effect of fake news on people's ability to process news information with their cognitive limitations and biases. Our approach was to build a cognitive model of an individual considering the apparent evidence and deciding who to support during 9 months leading up to the decisive election. We exposed the many copies of the model representing the population and its diversity to input representing the variety of reportedly news items. Over the period, we polled our synthesized electorate and compared the modeled population to available survey data. We start with a discussion of the data available for this phenomenon.

Data on News and Fake News

The fake news surrounding both political candidates originated from a variety of sources within and outside the US (Allcott & Gentzkow, 2017). Once these fake articles were published, they required social actors to promote the material to their peers (tweets/re-tweets on twitter or shares on Facebook). Researchers at Indiana University, Bloomington concluded that "bots", or algorithms that pose as social agents on social media, played a key role in the dissemination of fake news by tweeting and retweeting misinformation to promote some "news" items to a wider audience, and tagging popular users to project the appearance of relevance and legitimacy (Shao et al., 2017). With the help of these bots, any fake news article may go viral and possibly reach a large population of voting Americans. Importantly, human users consistently retweeted misinformation from bots, increasing the reach of fake articles.

In the last three months leading up to the election, shares of fake news articles on Facebook outnumbered those of real news (Silverman, 2016), see Table 1.

More specifically, shares of fake news articles were nearly four times more likely to be anti-Clinton/ pro-Trump than anti-Trump/pro-Clinton (Allcott & Gentzkow, 2017). This contrasts the coverage of real news, where overall the amount of pro-Clinton/anti-Trump articles slightly outnumbered pro-Trump/anti-Clinton articles (Patterson, 2016); see Table 2.

By compiling a database of real and fake articles in the three months leading up to the election and testing the average American's recollection of major headlines, Allcott and Gentzkow (2017) estimated that the average adult US citizen actually read and remembered 1.14 fake news articles. This does not account for the exposure to headlines and thumbnails that may operate more like political advertisements. In surveys conducted shortly after the election, those who considered Facebook a major source of news reported believing 83% of fake news headlines they

remembered seeing over the course of the election (Silverman & Singer-Vine, 2016).

Table 1: Engagement for Top 20 Election Stories on Facebook in 2016. Engagement refers to shares, reactions and comments on stories (Silverman, 2016).

	Feb-April	May-July	Aug-Nov 8
Mainstream News	12 mil	9 mil	7.3 mil
Fake News	3 mil	3 mil	8.7 mil

The ability to induce belief in misinformation is an important effect to investigate further, although it is difficult to quantify. Shao and colleagues (2017) reported a weak correlation between activity of bots claiming to be residents of individual states and differences between actual and predicted vote margins in the 2016 election for these states. While they were careful to note that this is in no way conclusive evidence that bots actually impacted the election results, it provides an opportunity to explore the subtle and unseen impact of bot-generated attitude change.

Cognitive Model Foundations

We started with the standard ACT-R cognitive architecture (Anderson, 2009) as a general architecture supporting the modeling an individual's cognition while performing the task of deciding how to interpret news items and developing a political opinion. We begin by considering the social factors that are likely to play a key role in developing an individual's political identity. After an individual holds firmly to a belief system, biased cognitive processes will lead to strengthened political identity and polarization over time (Lord et al., 1979). Our model specifically deals with how opinions are developed with repeated exposure, and strengthened as a result of motivated reasoning and cognitive biases (cognitive dissonance, belief bias and propaganda effect). We wanted to examine what experiences lead to the acceptance of fake news as factual information, as well as radicalization that can occur once an individual has a strong enough opinion.

Even when exposure to individual fake stories does not result in the adoption of an explicit belief, the accumulated effect can impact individuals' social cognitive processes in a number of ways. Viral news (real or fake) not only reaches a wide audience; it carries some social influence by virtue of the importance people place on observing and conforming to the majority opinions (Kiesler & Kiesler, 1969). If something seems to have a viral audience, without knowing how many bots shared it, people think that there is some group consensus that the information is valid.

In the absence of unbiased fact-checking and simply overwhelming news coverage, individuals are left on their own to determine the validity of any claims they encounter. "Epistemic vigilance" refers to the cognitive mechanisms that exist to constantly monitor the potential that we are being misled by another party (Sperber et al., 2010). Sperber et al.

Table 2: Positive and Negative Coverage for 2016 Presidential Candidates (Allcott & Gentzkow, 2017; Patterson, 2016).

	Negative Real	Positive Real	Negative Fake	Real Coverage
D	64	36	79.8	46.5
R	77	23	20.2	53.5

to preserve the value of communication by ensuring that information recipient can detect and punish dishonesty from the communicator.

Unfortunately, our mechanisms for epistemic vigilance seem to be tested by our frequent use of social media. In the cyber world, we lose valuable information such as paralinguistic cues that people use to evaluate honesty in face-to-face communication (Littlepage & Pineault, 1978), and the overall volume of information in online platforms overwhelms our attentional capacities, making it difficult to filter information by quality (Qiu et al., 2017). The delivery of computational propaganda disguised as peer-disseminated information exploits our handicapped epistemic vigilance mechanisms and threatens the overall usefulness of modern internet-based news disseminating platforms. Without adequate fact-checking or epistemic vigilance mechanisms, misinformation will be perceived and interpreted with biased cognitive processes which significantly contribute to the bipartisan divide in the US. Our model decided validity using the following cognitive mechanisms:

(1) "Motivated reasoning" refers to the phenomena where human decision making is impacted by emotional factors such as "cognitive dissonance", or the feeling of mental discomfort produced when experiencing information that seems to disconfirm an individual's beliefs (Festinger, 1957). Researchers have shown that liberals and conservatives experience cognitive dissonance in the same ways, preferring to listen to arguments supporting or opposing politicians or partisan topics that are in line with their own partisan identity to avoid discomfort (Frimer, Skitka & Motyl, 2017).

(2) The "belief bias" is the tendency for individuals to accept an argument as true if they think it is believable and in line with their identity rather than assessing the quality of the argument (Evans, Newstead & Byrne, 1993). When interpreting partisan arguments, for example, the belief bias would lead individuals to more easily believe information that is in line with their partisan identity completely independent of the quality of information. When surveyed, democrats and republicans were both about 15% more likely to believe

headlines of articles that appeared on social media in the three months leading up to the 2016 election when they fit with their partisan opinions (Allcott & Gentzkow, 2017), and this pattern could be stronger for people further isolated in partisan echo chambers.

(3) The "propaganda effect" refers to the tendency for people to rate information as more believable after they have been exposed to it previously (Begg et al., 1992). This effect works implicitly when the individual does not consciously remember the exposure, therefore highly partisan propaganda can leave an impression even if it is not consciously believed.

Cognitive Model Details

We use the standard, off-the-shelf ACT-R, version 7.3, without modifications to the architecture. The non-default parameters used turned on sub-symbolic computation (:esc T) and rule utility learning (:ul T) with a relatively low level of noise (:egs 0.25). We used the default for memory retrieval threshold.

Our model was based on the "choice" model of the ACT-R tutorial, which reads input presented on the screen and compares that information to memory to make a decision. Our model uses 12 rules to read and process the news inputs. A block diagram of the model is shown in Figure 1.

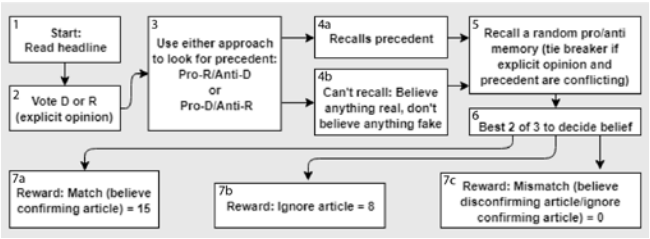


Figure 1. Citizen Model Block Diagram

The model begins a cycle by reading input in the form of a news item (fake or real, positive or negative, R or D), which the model perceives as a "chunk". Each news item is eventually either believed or disbelieved based on three pieces of information derived from proceeding productions: (1) The model expresses its explicit opinion in the form of an R or D vote based on the data it has collected since the first cycle. (2) Next, the model looks for a memory of a previous article that matches the current news item in order to find a belief precedent, deciding whether it wants to search for something that was previously believed or not believed given a specific candidate. If there is a retrieval error, the default precedent is that real news is believable and fake news is not. (3) A random exposure of positive or negative statements about a candidate is recalled in order to make a decision when the explicit opinion and belief precedent conflict. Initially, the modeled citizen had a random preference but it forms stronger opinions of the candidates over repeated exposure to election stories.

Rule utility learning was a part of the sub-symbolic representation of procedural knowledge of ACT-R. These rules became strengthened through use. Utility is considered to be a measure of the rule's value (Anderson, 2009). Models use rules with the highest utility. We used the same mechanism in our model to reward the results based on its utility. When the model ran, the production with the highest utility fired. The production firing had a reward value assigned to it. A reward value is propagated backwards through previous rule firings and depreciated by time.

To develop motivated reasoning, the modeled citizen experienced the highest reward when it was able to decide to believe an article that confirmed its explicit opinion (the vote; either pro-R/anti-D or pro-D/anti-R), resulting in a "match", and a medium reward if it could "ignore" disconfirming information. This was only possible if the citizen was able to recall a similar type of article that fit with the motivation (article precedent; belief bias). If the explicit opinion and the article precedent were contradictory, the citizen would make a "gut decision" based on some implicitly recalled information (propaganda effect). If the citizen was still unable to confirm their belief or ignore disconfirming information, it had to accept that the disconfirming information was true (attending a dissonant belief, resulting in 0 reward), resulting in a "mismatch". As the citizen was continuously exposed to news, it could attempt to reduce cognitive dissonance either by changing its explicit opinion about the candidates or by learning to strongly favor productions that were more likely to lead to matches and ignores.

Experiment

The experiment consisted of running a cognitive model representing an American citizen who was exposed to 1,000 news items over the last 9 months of the 2016 campaign cycle. Of those, 900 were randomly presented at the rates reported in Tables 1 and 2, with three phases of real and fake news ratios. To ensure that simulated citizens had some standardized experiences over time, an additional 100 real items that were presented at time points corresponding to some of the most influential real campaign stories. We recorded data over 100 runs of the model. The number of runs was arbitrary but intended to produce enough data to be useful.

Over the course of the experiment, the modeled citizen decided to believe or disbelieve each article, storing a memory of the event and slowly learning what candidate to support and what productions led to confirmation of that support and the least cognitive dissonance. Once the modeled citizen had a strong enough opinion about either candidate, it could partially or altogether stop believing real news that contradicted its current belief or believe fake news that confirmed its belief. This allowed us to investigate how motivated reasoning can affect truth-seeking behavior over

an election, possibly resulting in the adoption of radical opinions that are immune to the influence of facts.

In addition to the experiment described up until this point (which we will refer to in later sections as the "Troll" Condition), we also ran a version of the experiment in which the rate of fake news was kept constant over the 9 months (Feb-April in Table 1) and was equally disparaging of each candidate (deemed the "No Troll" condition), as well as a "No Troll" version that also had equal coverage of each candidate in real news ("No Troll/Equal Coverage"). These additional versions were added as control conditions against which we could compare the Troll results.

Finally, we ran 5 additional versions of the Troll condition to compare the rate of belief in real and fake news over time when the initial rate of belief in real ranged from 100% to 50% and fake ranged from 0% to 50%, in 10% intervals.

Experimental Results and Available Data

The experimental results are presented in two forms: as if our modeled citizens were regularly polled for their opinions and how the internal states of the model were shaped over time.

Polling trends were created by plotting the average explicit opinions over time for the 100 modeled citizens. Our model was able to reproduce polling trends by setting reward parameters and noise associated with learning such that a small amount of individual runs waffled between candidates on a significant amount of runs, but many explicitly supported the same candidate for the majority of trials. See Figure 2. See Figure 3 for comparison. The polling results for all 3 model conditions are compared to the actual electorate polling in Table 3.

The first internal factor we looked at was belief in real and fake news over time. Belief in fake news overtime was shown to increase. This is plotted as Figure 4.

Interestingly, the baseline model never disbelieved real news even though this was a possible (yet unlikely) outcome. To explore this further, the initial base rate of belief in real and fake news (retrieval failure, see box 4b in Figure 1) was varied from 100% to 50%. Results are shown in Figure 5.

While we explicitly set rewards to represent motivated reasoning, our model was still forced to observe disconfirming information (mismatch), thereby initially maintaining a solid representation of reality. However, the competition for attentional resources in the Troll condition resulted in decreased absorption of true information in the later months; see Figure 6.

Discussion

The issue of "fake news" had been a source of humor, but it now appears that fake news can affect the public's understanding enough to possibly change the outcome of a presidential election. The data on the frequency and type of fake news items circulated by the social media prior to the 2016 election was enough to cause our cognitive model of a US citizen to change the outcome of an election when averaged over 100 runs.

Our model shows that with the help of motivated reasoning, repeated exposure to large amounts of fake news results in competition for attentional resources that reduces the rate of

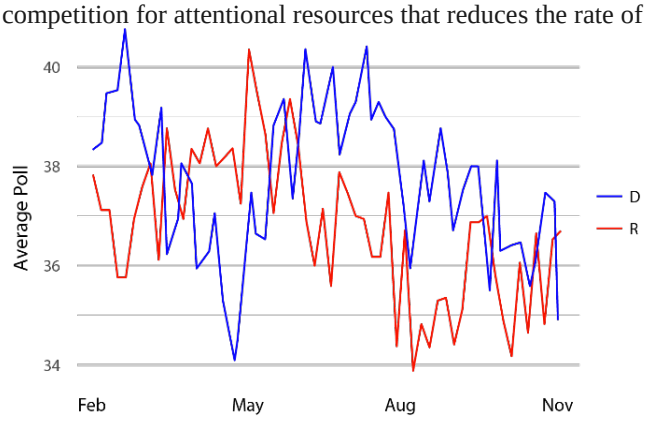


Figure 2. Simulated Polling Results Prior to the Election (averaged over 100 model runs) in the Troll Condition (Error bars are not shown due to variability of the mean).

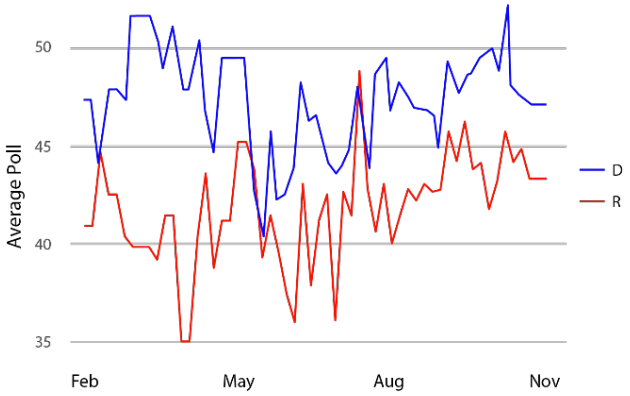


Figure 3. Electorate Opinion Polling prior to Election (Election G., 2016).

Table 3: True and Model Polling Results for Troll, No Troll, and No Troll/Equal Coverage. * indicates winner.

		Real Poll	Troll Poll	No Troll Poll	No Troll/Eq Cov Poll
Overall Average	D	**46.3 ***	**37.9 ***	*38.1 **	**41.3 ***
	R	41.7	36.8	*38.1 **	34.6
Last 5 Days	D	**46.3 ***	**36.5 ***	**38.9 ***	**42.7 ***
	R	42.7	35.8	38.6	33.9
Election Day	D	**48.2 ***	34.9	*39.5 **	**45.3 ***
	R	46.1	**36.7 ***	*39.5 **	34.5

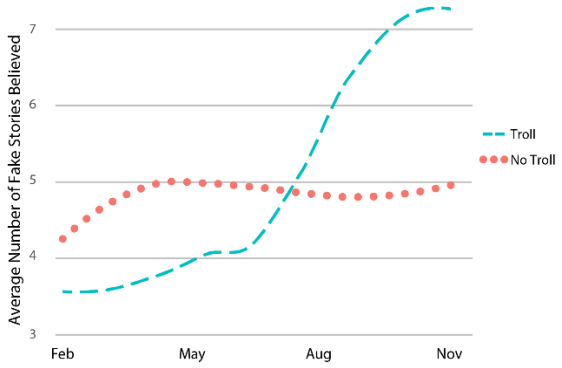


Figure 4. Modeled Citizen's Belief in Fake News in Troll and No Troll Conditions (averaged over 100 runs). Belief in fake news increases over time in the Troll condition.

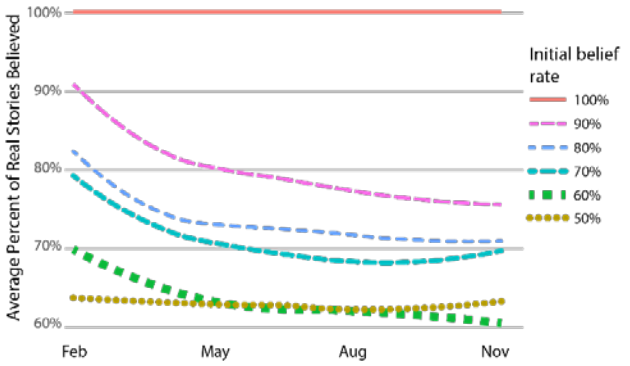


Figure 5. Average Percent of Real Stories Believed over 9 Months Beginning with Different Base Rates of Belief. Results are from the Troll Condition.

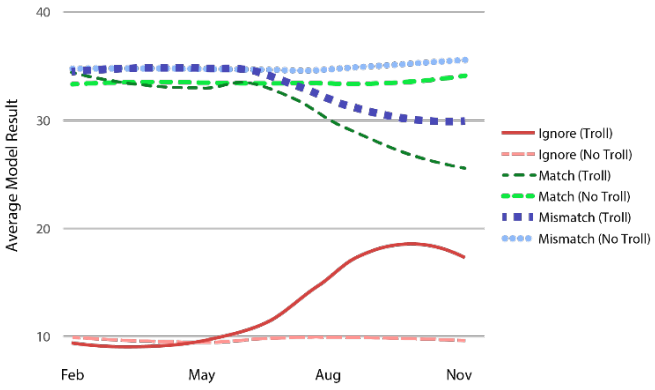


Figure 6. Model Results for Troll and No Troll Conditions. The increase in fake news results in a decline in the ability to process real information.

absorption of true information and increases the amount of fake news that is believed. This makes some sense, and the message is that a person's capability to process truth and update an opinion is hampered by the influx of fake news. Additionally, while there were more real anti-R/pro-D stories overall, the adoption of biases that were explicit (increased belief in fake news) and implicit (propaganda effect) against the heavily trolled candidate seemed to drive down the candidate's popularity in the Troll condition. Still, the impact of more real coverage for the R candidate also seemed to create more popularity for the R candidate in a type of "No press is bad press" fashion (see Table 3). While it would be encouraging to believe that real people never start to doubt true information, it is likely that people do not begin to develop an opinion with 100% truth detecting accuracy, and therefore some immunity to the truth can occur over time. The most dramatic impact occurs for those individuals who have the weakest discriminatory power before developing political bias (see Figure 5).

While our model was able to produce polling results that fit relatively well with the true polls (see Figure 2 vs. 3), there were a few limitations. We modeled our citizen to process about 10 news items daily, spread evenly over 24 hours per day, for the 9 months prior to the election. Our model begins with no prior political identity or opinion of either candidate. The average real citizen would likely have had some political identity before the 2016 election cycle, which tends to lead people to surround themselves with like-minded individuals, which would have affected their true rate of exposure to partisan stories (real and fake). Additionally, our model only understood these stories as simplified chunks, not paying attention to the language of a headline, the user who posted it or the source that published it, which are all pieces of information that would enter into the consideration of validity. Future work will seek to address these processes.

This work is an example of the type of modeling possible in the field of computational social science, where models of individual agents reacting to their environment and other agents can demonstrate possible macro-level results from relatively simple micro-level agents. Combining cognitive modeling and computational social science improves the credibility of results.

Acknowledgments

This work was supported in part by the Center for Social Complexity at George Mason University. The opinions, findings and conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of the sponsors.

References

Allcott, H., & Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2), 211-36.

Anderson, J. R. (2009). *How can the human mind occur in the physical universe?*. Oxford University Press.

Begg, I. M., Anas, A., & Farinacci, S. (1992). Dissociation of processes in belief: Source recollection, statement familiarity, and the illusion of truth. *Journal of Experimental Psychology: General*, *121*(4), 446.

Election, G. Trump vs. Clinton. *RealClear Politics. URL: http://www.realclearpolitics.com/epolls/2016/president/us/general_election_trump_vs_clinton-5491. html*.

Evans, J. S. B., Newstead, S. E., & Byrne, R. M. (1993). *Human reasoning: The psychology of deduction*. Psychology Press.

Festinger, L. (1962). *A theory of cognitive dissonance* (Vol. 2). Stanford university press.

Frimer, J. A., Skitka, L. J., & Motyl, M. (2017). Liberals and conservatives are similarly motivated to avoid exposure to one another's opinions. *Journal of Experimental Social Psychology*, *72*, 1-12.

Kiesler, C. A., & Kiesler, S. B. (1969). *Conformity*. Addison Wesley Publishing Company.

Littlepage, G., & Pineault, T. (1978). Verbal, facial, and paralinguistic cues to the detection of truth and lying. *Personality and Social Psychology Bulletin*, *4*(3), 461-464.

Lord, C. G., Ross, L., & Lepper, M. R. (1979). Biased assimilation and attitude polarization: The effects of prior theories on subsequently considered evidence. *Journal of personality and social psychology*, *37*(11), 2098.

Patterson, T. E. (2016). News coverage of the 2016 general election: How the press failed the voters.

Qiu, X., Oliveira, D. F., Shirazi, A. S., Flammini, A., & Menczer, F. (2017). Limited individual attention and online virality of low-quality information. *Nature Human Behaviour*, *1*(7), 0132.

Shao, C., Ciampaglia, G. L., Varol, O., Flammini, A., & Menczer, F. (2017). The spread of fake news by social bots. *arXiv preprint arXiv:1707.07592*.

Silverman, C., & Singer-Vine, J. (2016). Most Americans who see fake news believe it, new survey says. *BuzzFeed News*.

Silverman, C. (2016). This analysis shows how viral fake election news stories outperformed real news on Facebook. *BuzzFeed News*.

Sperber, D., Clément, F., Heintz, C., Mascaro, O., Mercier, H., Origgi, G., & Wilson, D. (2010). Epistemic vigilance. *Mind & Language*, *25*(4), 359-393.

A Neural Field Model of Word Recognition

Andrew P. Valenti (andrew.valenti@tufts.edu)

Bradley Oosterveld (bradley.oosterveld@tufts.edu)

Matthias Scheutz (matthias.scheutz@tufts.edu)
Tufts University Human-Robot Interaction Laboratory, 200 Boston Ave.
Medford, MA 02155

Abstract

We show how temporal and spatial information can be represented as stable patterns in a dynamical system. We describe a model in which category perception arises from the incremental recognition of temporal patterns from sequences of inputs and this is accomplished by decoding a pool of recurrently connected artificial neurons which is called a neural field. In an example application, we use these patterns to identify a set of words which share the word onset represented by the input sequence, consistent with the Marslen-Wilson COHORT model of word recognition. Similarly, we evaluate the extent to which information contained in the bottom-up sensory signal can be used to determine word boundaries. We suggest it is plausible that a neural field offers a naturalistic explanation of how perception arises in word processing.

Keywords: dynamic field theory, neural fields, connectionist model, word recognition, COHORT model, machine learning

Introduction

The brain encodes and processes sensory input acquired from the environment. Sensory input, regardless of modality, is encoded as spatiotemporal patterns, and a superior form of pattern processing has evolved in humans coinciding with the expansion of the neocortex. In this brain structure, several essential cognitive processes such as visual, auditory, and speech perception occur (Koch, 2004; Mattson, 2014). These processes include not only recognizing patterns, but also classifying them (Grossberg, 2005). During this processing, different sensory inputs which represent members of the same category are mapped to a singular representation for that category. In speech processing, for example, all pronunciations of the phoneme "ə", are mapped to the same pattern, allowing for invariance in speech perception across multiple speakers (Kleinschmidt & Jaeger, 2015). Consistent with these hypotheses, our model uses patterns of activation to represent sequences of states in the context of perceiving words; we modeled these states as equilibriums in a *neural field*.

The human neocortex consists of six layers of tissue containing approximately 10^{10} neurons. Columns of tissue can be represented mathematically as neural fields, which form patterns of activation through interaction with each other (Amari, 1977). These interactions between fields generate patterns of activation in a fashion that is believed to be similar to how sensory information is represented in the human neocortex (Amari, 1977; Brady, 2012). These patterns represent an encoding of spatial and temporal information from the brain's sensory input stream.

Each neuron in a neural field F (Figure 1) is connected to each of its neighbors with weights that create an on-center off-surround activation pattern, where the closest neighbors provide a positive influence on activation, further neighbors a negative influence, and the furthest no influence. If given no input and random initial conditions, the units of the field are guaranteed to quickly fall into a stable equilibrium state. Different equilibrium states of a field can be associated with different inputs, and thus the states of activation in a neural field can be used to store information by associating them with category labels (Valenti, Brady, Scheutz, Holcomb, & Pu, 2016).

In this work, we demonstrate a model of word perception using neural fields. Our research is not focused the initial interaction between perceptual signals and the sensory apparatus. We are instead interested in the processing of the output of such apparatuses, and how it can be used to constrain the patterns of activation in higher level cognitive processes, like lexical representation. Our model uses two neural fields, each representing a level of cognitive processing. Since sensory information unfolds over time as a continuous sequence, the input presented to the first neural field is a sequence of feature vectors which represent the letters of an artificial font. Sequences of output features from the first field representing letters are then presented as input to the second field which identifies likely word boundaries and classifies these letter sequences as words.

There are many theories about how patterns of activation in the lexicon are formed once the sensory information has been received (Dahan & Magnuson, 2006). This work focuses on the Marslen-Wilson (1987) COHORT model, which theorizes that information contained in the bottom-up perceptual signal can be exploited to determine which lexical items should be activated, and also used to identify perceptual characteristics such as word boundaries. To explore the extent to which this information is sufficient, we have developed a model where word onsets constrain the set of activated lexical entities such that word onsets activate lexical items with shared onsets. Our model thus makes predictions similarly to the COHORT model; the initial information contained in the sensory signal influences the activation of an initial word-cohort, allowing it to predict word boundaries in a higher level of processing.

Representing State with a Neural Field

Our model is composed of two layers of neural fields. The structure of a single layer is shown in Figure 1. An Input vector (I) is fully connected to the neural field (F) by input

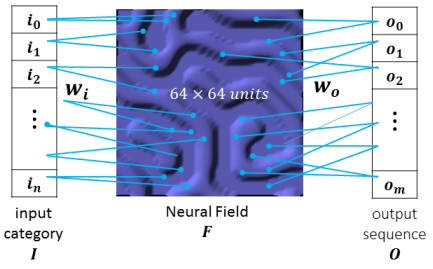


Figure 1: Single field design. Note: I and O are fully connected to F, but only a few connections are depicted here, for clarity.

weights (W_i). F is fully connected to an output vector (O) by output weights (W_o). In our model, such a layer (input, field, and output) can be interpreted as a cognitive processing layer, computing a specific function such as letter or word detection. These layers can be combined to represent a hierarchy of cognitive processes shown in Figure 3.

Our model is based on the following principles of dynamic field theory. Patterns can be stored as stable equilibrium states. A sequence can be "remembered" as a unique equilibrium, unrelated to any previously generated equilibrium, by calculating the sum of the pattern generated by the current input and the pattern representing the previous sequence. Fields converge to a stable equilibrium state after applying a finite series of "settling" operations after which the field ceases to change. Finally, fields can be forced out of a stable state into a target state by applying a finite series of operations which includes the target as its field input.

Field Dynamics

The input to a layer is received as a vector I, whose dimensionality is the number of discrete categories in the input domain. This input is used to first calculate the $n \times n$ matrix F_t (in our evaluation $n = 64$), which represents field activation at the given point in the sequence of input vectors. F_t is calculated using the following equations which are a variation of those widely used in dynamical systems (Amari, 1977).

$$D = W_i I \tag{1}$$

$$\sigma(x) = \begin{cases} \dfrac{x}{x+1} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2}$$

$$S = W_{mh}\ \sigma(F_{t-1} + D) \tag{3}$$

$$F_t = \sigma(S + h + n) \tag{4}$$

As shown in Equation (1), the driver input, D, is generated by multiplying the input vector I by the input weights, W_i (dimensionality $n \times n \times \| I \|$). D is then added to the current field equilibrium, F_{t-1}, and the result is squashed to the range $[0,1]$ using Equation (2). This new equilibrium represents the sequence of input seen up to time t, plus the input at t. The result is multiplied by the within field weights, W_{mh}, which are defined using the Mexican hat function;[1] the result is the field influence term, S, Equation (3). Small bias h and noise n terms are added to the field influence, and the result is squashed again to produce the field activation F_t, Equation (4).

Settling to an Equilibrium State

Once a field has been updated from an input, a settling operation is applied resulting in convergence to an equilibrium state. This process is expressed in Equation (5), which is based on Equations (3) and (4) with the notable exception that the value of the field at the previous time step is not added to the input. This operation is repeated until a stable equilibrium is reached, determined by comparing the field activation at time t with its activation at $t-1$, repeating until the difference is below a small epsilon value.

$$S = W_{mh}\ \sigma(S + n) \tag{5}$$

Layer Output

The settled equilibrium is used to calculate the layer's output vector, O, whose dimensionality is the number of categories in the output domain.

$$O = tanh(F_t W_o) \tag{6}$$

The state of the field is multiplied by the output weights W_o ($n \times n \times \| O \|$), and their product is passed through the hyperbolic tangent activation function. The result is a vector whose values are normalized to the range $[-1,1]$.

The COHORT Model Using Neural Fields

The Marslen-Wilson (1987) COHORT model of spoken word recognition suggests that the real-time constraints of a speech signal influence how bottom-up information is used to determine which items in the mental lexicon become activated. According to the model, on each new input onset only the cohort of possible values remains activated; a cohort is the set of all lexical items that share an onset. A decision is reached only when one possible value remains in the cohort. Consistent with the original version of COHORT (which assumed a highly categorized, abstract string of phonemes rather than feature vectors as input), our model uses only the features present in the bottom-up sensory input to develop the cohort.

The neural fields in our model simulate what happens when sensory information makes contact with the mental lexicon. It is assumed that, once past the sensory apparatus, information flow through differentiated neural architectures, e.g., visual or auditory, is represented in the same fashion. In our evaluation we chose to focus on visual information, and the following sections describe how it is handled by the model.

[1] A Mexican Hat function where D is the Euclidean distance between two units in the field: $W_{mh} = e^{\frac{-D^2}{r^2 d}} \cdot (cos(\frac{\pi D}{2r}) - z) \cdot (\frac{1}{1-z}), r = 4.5, z = 0.15$

Model Input

The input to our model is a sequence of feature vectors which represent the output of the visual perceptual system. As in the Interactive Activation model of word recognition (McClelland & Rumelhart, 1981), visual features are extracted from the raw input, sequences of letters, by separating each letter into a set of component features. These features can be thought of as the pen strokes used to write the letter. For simplicity, we have chosen the font used by Rumelhart and Siple (1974) which is shown in Figure 2. Sequences of feature vectors are generated from a letter by arbitrarily circumnavigating the font clockwise from the outermost feature, spiraling inward. For example, the letter "R" is represented by the sequence [0, 1, 4, 5, 8, 9, 12].

In our implementation, the model receives visual features as input. As mentioned earlier, the input features could also represent information from other sensory modalities. For example, the input features could also come from a speech recognizer and instead represent sequences of phonemes. At the cognitive processing level, the model's basic results do not depend on what the input represents. With different input features there are obviously low-level feature processing issues (e.g., different types of variance in the input) which are outside the scope of this work.

(a) Labeled Segments (b) Letters

Figure 2: 14-segment display using the letter font.

Model Architecture

The architecture of our model is shown in Figure 3. It uses two neural fields, each representing a stage of cognitive processing: *F1* which is a letter detector, and *F2* which is a word detector. $F1$ and $F2$ are connected via a Send Gate which controls when information from $F1$ is sent to $F2$.

Before the model can be used, it must first be initialized. This initialization generates the associations between input and neural field equilibrium states which are used to detect sequence boundaries. During this initialization the model is presented with sequences of input features and the boundaries between them that represent meaningful units. Perceptrons are trained to detect these boundaries based on the equilibrium states which represent them.

Once initialized, information flows through the model as shown in Figure 3. Letter segment sequences previously generated by the visual recognizer flow as input to $F1$; letters detected by $F1$ flow as letter sequences to the word detector $F2$. Perception arises as the generated pattern predicts a set, or cohort, of letter or word candidates. As new features are fed incrementally into the model, a new pattern is generated and each field's perceptron updates its prediction, removing candidates from its cohort. Letter or word recognition occurs when there is a single candidate left in the respective cohort. When recognition occurs, the send gate is opened sending the output perceptron's value to the next layer as input. In the case of the letter layer, the output is sent as input to the word layer; in the case of the word layer, the output value of the perceptron is used by the decoder perceptron to generate results interpretable by a human.

Model Initialization

The characteristic theory of the COHORT model is instantiated in the neural field model during initialization. This initialization forms the associations between input features and neural field equilibrium states used during perception. This initialization is composed of three steps: (1) Initial equilibrium generation (2) W_i training (3) W_o training.

The initial values of the weight matrices used in the model ($F1$: W_i, W_o and $F2$: W_i, W_o) are chosen randomly from a truncated normal distribution with a standard deviation of: $\frac{2}{\sqrt{n_{inputs}}}$ Using this particular standard deviation helps the training to converge more quickly (Géron, 2017).

Initial Equilibrium Generation: A "seed" equilibrium is generated to represent each unique input feature a field will receive. For the letter detector, $F1$, 15 equilibriums are generated to represent each of the 14 possible letter visual features, plus an equilibrium to represent the beginning of a sequence when no input has been presented yet. For the word detector, $F2$, 27 equilibriums are generated, for the 26 letters in the English alphabet and one more for the initial sate. These initial equilibriums are generated by a variation of Equation 1 where I is the product of a one-hot vector whose 1-bit corresponds to the ordinal value of the letter segment in the range $[0,15]$ or letter in the range $[0,27]$ and the randomly drawn W_i for the given field.

W_i Training: For a feature vector (i.e., letter segments for the first field, letters for the second), a set of weights (W_i) is trained which will reliably reproduce the initial equilibrium associated with that feature vector. The model's operation assumes that a *settled* field generated from new input can be added to the current settled field to produce a new equilibrium representing the input sequence seen thus far. Without trained driver weights, an *unsettled* equilibrium would be added, violating a core model assumption. The training of W_i, uses a version of the perceptron learning rule, Equation 7, to train the single layer perceptron whose activation is found by multiplying the driver input by W_i,

$$\Delta W_i = \eta I(Target - IW_i) \tag{7}$$

where *Target* is the seed equilibrium for the category and η is a learning rate. Training proceeds until $Target - IW_i < 0.0001$. This approach is a variation of Hebbian learning, a biologically plausible mechanism for learning associations

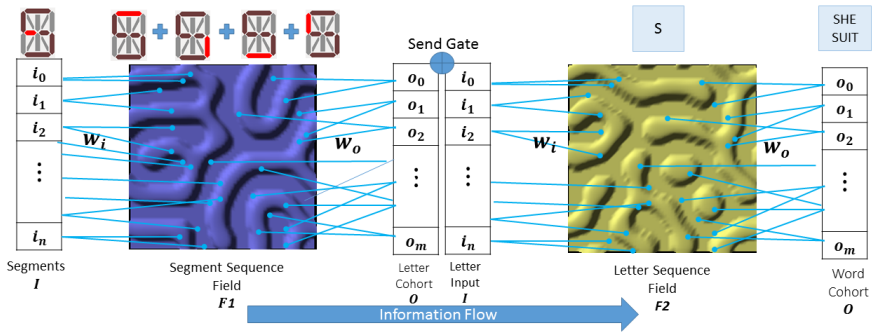


Figure 3: Neural Field based Cohort Model architecture.

between neurons (Laszlo & Plaut, 2012).

W_o **Training:** The output weights W_o map a field's current equilibrium to the output domain relevant to its cognitive layer (i.e., letters cohort or words cohort). For a given cognitive layer the output vector O represents the members of the cohort that are currently active. O is the size of the lexicon of known labels at the given cognitive level, each of its elements representing a member of the lexicon. A member of the lexicon is considered activated if the value of its corresponding element in O is above an activation threshold. For example, the letter segment sequence [0, 1] is the prefix of the letters {A, B, D, O, P, Q, R} (see Figure 2a).

The weights W_o are trained so that the same value of O can be calculated every time a corresponding equilibrium is present in F. The weights are updated using Equation 8,

$$\Delta W_o = \eta F(Target - FW_o) \tag{8}$$

where $Target$ is the vector in the output domain (e.g., letters or words) indicating cohort membership of the lexical entries whose onset is represented by the equilibrium of the neural field F. Training proceeds until $Target - FW_o < 0.001$.

Detecting a New Sequence

An equilibrium for a sequence is generated by adding the seed equilibrium of the new element of the sequence to the current sequence equilibrium. For the first segment in a letter, its seed equilibrium is added to a default value (i.e., the 15^{th} seed equilibrium). This process is repeated for each segment of the letter and after each addition, the field is settled. A challenge in implementing the COHORT model is detecting when a new sequence begins. In our model, an input sequence is "remembered" as an equilibrium whose value is the sum of the equilibriums seen as input thus far. Thus, when the end of the sequence is detected there must be some way of resetting the field so that the next sequence is not affected by the previous sequence. To do this, each field has a reset gate whose purpose is to detect the conditions under which the field should be reset to its default equilibrium.

The default equilibrium is the starting state to which subsequent equilibriums are added. The model hypothesizes that a reset signal represents constraints arriving top-down from higher cognitive processing levels (e.g., syntactic, semantic, pragmatic) as well as bottom-up from the features contained in the input data.

The Send Gate

Each layer of the model has an associated Send Gate which controls the information that it sends to the next highest level of cognitive processing. The first layer's send gate connects the letter detector field to the word detector field and the second layer's determines the overall output of the model. In different configuration of the model, the second layer's send gate could connect to a third field and so on. Send Gate processing is the same for every layer (refer to Figure 4). First, the input features A are presented, and the field is updated B. The cohort is then calculated C and evaluated by the Reset Gate D. The field may or may not then be reset to its default state. The cohort is calculated and if it has shrunken to one member, the Send Gate E opens.

Notice that the Send Gate only opens when the cohort has shrunk to one member. Thus we must ensure that the state of the cohort is reset so that a new sequence can be subsequently recognized,otherwise a feature that is repeated across category boundaries will not be recognized. The Send Gate behavior models the recognition point prediction of the COHORT model which states that word recognition occurs as soon as sufficient information is received such that all other candidates are eliminated (Marslen-Wilson, 1987).

Model Evaluation

The primary goal of our research was to determine whether neural fields are a plausible way to model word perception. Prior research theorizes that humans represent word forms as categories, abstracted away from variability (Dahan & Magnuson, 2006) and it is this view that our model seeks to explore. There are several well-known cognitive models (e.g., COHORT, TRACE, Neighborhood Word Activation) whose theories make different predictions once the input signal make contact with the lexicon. We chose the COHORT model as a starting point and explore whether two of its predictions can emerge from our neural field model: word-initial cohort and the identification of word boundaries. The operation of the model is summarized as follows:

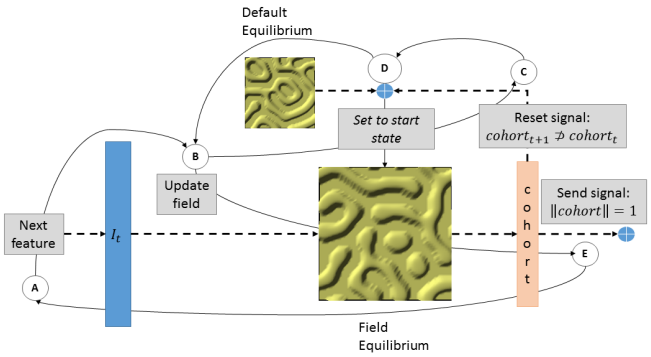


Figure 4: Reset/Send Signal Processing. If the reset gate (D) is open, it will set the field to a default equilibrium; otherwise it will update the field to the sum of the equilibrium of the current field and the equilibrium of the new feature. Send gate processing (E) takes place *after* the reset gate is processed.

1. Each feature (i.e., letter segment) of sensory input is converted to a pattern.
2. Sequences are generated by adding the current input's pattern to the previous input's pattern.
3. Perception arises as the generated pattern predicts a set, or "cohort", of letter or word candidates. A perceptron is trained to decode the pattern and interpret the prediction.
4. As new features are fed into the model, a new pattern is generated and the perceptron updates its prediction, removing candidates from the cohort.
5. Letter or word recognition occurs when there is one candidate left in the cohort.
6. New categories are recognized at the point when either the cohort is empty or when new candidates are added.

Data

The TIMIT corpus (Garofolo, Lamel, Fisher, Fiscus, & Pallet, 1993) provides a set of 10 phonetically rich sentences spoken by 630 speakers of eight major dialects of American English which are annotated at the word and phoneme level. The annotations of the corpus were used as a set of naturally occurring sequences to train the model's letter and word detectors. The text of the corpus was used to create feature vectors, as described in *Model Input*, which was presented to the $F1$ as a sequence of letter segments.

Results

The entirety of the TIMIT training set was pre-processed by the visual feature recognizer and its output, an unbroken sequence of letter segments was presented as input to the model. In the first experiment, the model was artificially reset to a default state at the end of every word so that errors in the perception of one word did not affect the perception of other words. This was done to verify correct operation of the model. For 100% of the words in the lexicon, the activation matched the ground truth for every letter segment in that word. Furthermore the model generated the correct cohort (when one existed) of letters for every letter segment sequence and of words for every letter sequence. In a separate experiment (Valenti, Oosterveld, & Scheutz, 2017), the model was not reset and in 82.5% of cases, the model detected the word level transition, suggesting that bottom-up information alone is insufficient to detect word boundaries.

Discussion

The model uses the structure of the data to represent top-down cues which are simulated through a "forced reset" when the start of a new letter or word is detected from the structure of the input data set. This is not ideal but allows the model to continue processing when the bottom-up cues alone are insufficient. One alternative to the forced reset would be to train a detector to recognize likely word boundaries in a training corpus, using it to augment the existing cohort-based reset mechanism. Consider the following sequence of letters without any explicit separation (we could have equally used a letter segment sequence, but that would have been harder to visualize):

shewashedyourdarksuitingreasywashwaterallyear

Humans can usually distinguish each letter sequence of a word and consequently recognize each word of the target sentence; however it is not as straightforward for a computer model to do so. Without further information constraints, a naïve model might correctly reject all sequences of letters that form non-words (e.g., *shew*) but erroneously recognize legal words such as *suiting*, resulting in a syntactically implausible reading of the sentence. Our model attempts to discern sequence boundaries by exploiting the cohort dynamics when processing letters. As a sequence of letters is read into the model, a cohort of possible words is initially formed which shrinks in size until only a single word candidate is left; this is the word's recognition point. If a shrinking cohort begins to grow again when a new letter is added to the sequence,decided this might indicate the start of a new word sequence and that the model should reset the field (*D* in Figure 4).

Since the present design does not model higher level cognitive processes, we abstract over all those that might be relevant to detecting a category boundary and combine them into one signal per field called *forcedReset*. Specifically, the data is preprocessed by the visual feature recognizer so that the letter segments have been grouped into sequences by letter. This roughly corresponds to how the higher areas of the visual cortex constrain lower area feature sequences during perception (Friston, 2005). The model uses this information to force the letter detector field to reset at the start of every new sequence. Likewise, the model uses the word size as a top-down cue to force a reset in the $F2$ word detector. During evaluation, the percentage of times the model accurately detects a word boundary using only the bottom-up signal is calculated; the

forced reset ensures the model can continue processing when there is insufficient bottom-up information.

Future Directions

The first version of COHORT assumed input to be an abstract phoneme string. Thus, we arbitrarily chose to present visual input to the neural field as an unambiguous, noiseless sequence of letter segments which made it easier to visualize the model's operation in its graphical user interface. Real-world data is noisy yet perception still arises from these cognitive "noisy channels". Developing a design that incorporates noisy channels is key to understanding situated cognitive processes. Similarly, the input is invariant. In the speech perception domain, humans can usually recognize what is being said regardless of the speaker's accent, gender, etc. The model design needs to incorporate the ability to map varying input to invariant representations in order to simulate human performance in most perception domains. The model uses bottom-up information contained in the input signal to determine word boundaries, which is insufficient for 100% accuracy. Training an additional perceptron on a large speech corpus such as TIMIT should allow the model to statistically learn when a word boundary is likely to occur and this can be as a top-down cue to be added to the reset signal and improve its accuracy. Lastly, human cognitive language processing in the auditory and visual domains is often studied using electro physiological measures such as Event-related Potentials (ERPs). We have previously demonstrated a mapping of a single neural field model's dynamics to an ERP component (Valenti et al., 2016). Extending this to multiple fields could lead to models of human cognitive performance under varying cognitive workloads.

Conclusion

We explore the cognitive process of word recognition by creating a dynamacist model of the COHORT theory of Marslen-Wilson (Marslen-Wilson, 1987). This theory describes how sensory input is mapped to a specific word from a person's mental lexicon. Whereas Marslen-Wilson predicted the identification of a word cohort from which a unique word is selected and recognized he did not address how it might arise functionally from the input signal nor did he specify an implementation of the model. Moreover, we know of only one implementation of COHORT (Johnson & Pugh, 1994); it too conceives of encoding the input as patterns from which a cohort emerges and resolves. However it does not discuss the underlying algorithm for this process nor how it was trained, so it is difficult to assess its plausibility. In contrast, the presented model provides a general way to encode sequences in patterns and to find positions within those sequences which is applicable to any type of sensory information unfolding over time.

References

Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological cybernetics*, *27*(2), 77–87.

Brady, M. C. (2012). *A field-based artificial neural network with cerebellar model for complex motor sequence learning*. Unpublished doctoral dissertation, Indiana University.

Dahan, D., & Magnuson, J. S. (2006). Spoken Word Recognition. In M. J. Traxler & M. A. Gemsbacher (Eds.), *Handbook of Psycholinguistics.*

Friston, K. (2005). A theory of cortical responses. *Philosophical Transactions of the Royal Society London, Series B, Biological Sciences*, *360*(1456), 815–836.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., & Pallet, D. S. (1993). DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM, NIST speech disc 1-1.1. *NASA STI/Recon technical report n*, *93*.

Géron, A. (2017). *Hands-on Machine Learning with Scikit-Learn & TensorFlow* (1st ed.). Sebastopol, CA: O'Reilly.

Grossberg, S. (2005). Adaptive resonance theory. In L. Nadel (Ed.), *The encyclopedia of cognitive science* (1st ed.). Wiley.

Johnson, N. F., & Pugh, K. R. (1994). A cohort model of visual word recognition. *Cognitive Psychology*, *26*, 240–346.

Kleinschmidt, D. F., & Jaeger, T. F. (2015). Robust speech perception: Recognize the familiar, generalize to the similar, adapt to the novel. *Psychological Review*, *122*(2), 148–203.

Koch, C. (2004). *The Quest for Consciousness: a Neurobiological Approach* (1st ed.). Englewood, CO: Roberts and Company Publishers.

Laszlo, S., & Plaut, D. (2012). A neurally plausible parallel distributed processing model of event-related potential reading data. *Brain and Language*, *120*, 271–281.

Marslen-Wilson, W. D. (1987). Functional parallelism in spoken word-recognition. *Cognition*, *25*, 71–102.

Mattson, M. (2014). Superior pattern processing is the essence of the evolved human brain. *Frontiers in Neuroscience*, *8*(265). doi: 10.3389/fnins.2014.00265

McClelland, J., & Rumelhart, D. (1981). An Interactive Activation Model of Contextual Effects in letter perception: Part 1. *Psychology Review*, *88*(5), 375–407.

Rumelhart, D., & Siple, P. (1974). Process of recognizing tachistoscopically presented words. *Psychology Review*, *81*, 99–118.

Valenti, A. P., Brady, M. C., Scheutz, M. J., Holcomb, P. J., & Pu, H. (2016). A neural field model of word repetition effects in early time-course ERPs in spoken word perception. In A.Papafragou, D. Grodner, D. Mirman, & J. Trueswell (Eds.), *Proceedings of the 38th Annual Conference of the Cognitive Science Society* (pp. 2765–2770).

Valenti, A. P., Oosterveld, B., & Scheutz, M. J. (2017, Sep. 6-8). A neural field model of sequence perception. In *Cognitive Computational Neuroscience Conference 2017.* New York, NY.

Modelling the Effect of Depression on Working Memory

Maarten A. van der Velde (m.a.van.der.velde@rug.nl)
Marieke K. van Vugt (m.k.van.vugt@rug.nl)
Niels A. Taatgen (n.a.taatgen@rug.nl)
Department of Artificial Intelligence, University of Groningen
Nijenborgh 9, 9747 AG Groningen, The Netherlands

Abstract

Individuals with depression are prone to engaging in rumination, a process in which attention turns inwards to narrowly-focused, negative patterns of thought, at the cost of attending to a task. Other core deficits associated with depression are weaker inhibition of information that is no longer relevant, and a negative perceptual bias. Here, we present a computational cognitive model that uses these mechanisms to explain performance on an n-back task in which the stimuli are faces with different emotional expressions, and in which depressed participants exhibit specific impairments. These impairments are explained by assuming that depressed participants selectively elaborate on sad items as they are removed from working memory, and that they have a perceptual bias towards sad faces. In this way, by specifying a mechanism by which performance impairments come about, the model helps to provide a deeper understanding of the cognitive processes underlying behaviour.
Keywords: depression; rumination; mind-wandering; working memory; computational cognitive modelling.

Introduction

Depression is an important mental health issue, but much is still unknown about its cognitive mechanisms. A major cognitive component of depression is thought to be rumination, which can be conceptualised as a maladaptive form of mind-wandering (Marchetti, Koster, Klinger, & Alloy, 2016). Mind-wandering, a process in which attention is directed away from a task towards internal thoughts and memories, can take up as much as 50% of our waking hours, and can have both positive and negative effects on cognitive performance (Mooneyham & Schooler, 2013). What sets rumination apart from other forms of mind-wandering is that it is characterised by a thematic narrowness, focusing primarily on negative memories and thoughts related to one's current dysphoric state. Ruminators tend to repeatedly engage in certain narrow trains of thought from which it can be difficult to escape (Christoff, Irving, Fox, Spreng, & Andrews-Hanna, 2016). While both depressed and non-depressed individuals have been observed to engage in off-task thinking during laboratory tasks (e.g., Smallwood, Obonsawin, & Heim, 2003), participants with depression spend more time mind-wandering than their non-depressed counterparts (Hoffmann, Banzhaf, Kanske, Bermpohl, & Singer, 2016).

Rumination has been associated with several cognitive effects, including inhibitory and switching deficits. Individuals who ruminate tend to have more difficulty inhibiting information that is no longer relevant to their current situation, compared to non-ruminators. A study by Whitmer and Banich (2010), in which participants memorised a list of study items and subsequently rehearsed only a subset of this list, found that those with a tendency for rumination inhibited unrehearsed items from the original list less strongly than non-ruminators. This inability to effectively inhibit outdated information also affects working memory. In a task-switching paradigm, the occurrence of depressive rumination is associated with a weaker inhibition of previous task sets (Whitmer & Banich, 2007). Depressed individuals find it particularly difficult to inhibit obsolete information when it has a negative valence (Joormann & Gotlib, 2008), whereas non-depressed individuals have similar difficulty inhibiting positive stimuli (Deveney & Deldin, 2006), suggesting a mood-congruency effect.

Aside from the inhibitory deficits associated with rumination, depression can also influence visual perception and attention. Whereas healthy humans commonly exhibit a perceptual bias towards positive stimuli, perception in those with depression tends to be coloured by a negativity bias. When Gollan, Pane, McCloskey, and Coccaro (2008) asked participants to judge the emotion of a neutral face in a forced-choice task, non-depressed participants saw its emotion as happy more often than sad, while depressed participants were more inclined to judge the same face as sad. Visual attention is also not immune to bias: depressed individuals tend to demonstrate selective attention towards sad faces and/or away from happy faces (Bourke, Douglas, & Porter, 2010; Suslow, Junghanns, & Arolt, 2001). Indeed, depression may be linked to a general negativity bias, which also reveals itself in, e.g., memory recall (Dalgleish & Watts, 1990).

Gaining a better understanding of how depression influences and interacts with cognition could potentially aid the development of better diagnostic tools and more effective treatments. Schemes that are currently used to diagnose psychiatric disorders tend to be overly simplistic in their categorisation, and do not translate well to the clinical level (van Os et al., 1999). Existing cognitive theories of depression, generally formulated as verbal accounts, can be difficult to generalise or integrate with other theories. In recent years, the field of computational psychiatry has demonstrated the benefits of striving towards more rigorous accounts of the role of cognition in mental illness (Adams, Huys, & Roiser, 2016).

We have been developing such a computational model in the ACT-R cognitive architecture (Anderson, 2007), which is well-suited to translating verbal theories into quantitative predictions. Increasing the prominence of negative items in the model's declarative memory transforms its regular mind-wandering behaviour into more persistent, negatively-coloured depressive rumination (van Vugt, van der Velde, &

ESM-MERGE Investigators, 2018). When applied to a sustained attention task, the model predicts lower response accuracy as a result of this change.

Here, we extend this model to a more complex task, the *emotional n-back* task (Levens & Gotlib, 2010). Modelling this task requires the addition of some mechanisms to our existing model. The task, a modified version of the *n*-back, uses images of happy, neutral, and sad faces as its stimuli. It involves complex working memory operations, as participants are required to maintain an up-to-date mental list of recently seen items in their memory in order to respond correctly (Lovett, Daily, & Reder, 2000; Juvina & Taatgen, 2007). Because of its use of emotional face stimuli and its reliance on working memory, this task is expected to elicit many of the cognitive effects of depression.

Levens and Gotlib (2010) had depressed and non-depressed participants perform the emotional *n*-back task, and found that, while depressed participants responded more slowly and slightly less frequently across conditions, they were as accurate as healthy controls at identifying whether a face had the same expression as one shown two trials earlier. Response times did reveal differences in working memory updating: depressed participants responded relatively quickly in trials with a sad stimulus, compared to happy or neutral trials, indicating easier integration of a sad face into their mental list. Furthermore, depressed participants were relatively slow when breaking a former set of matching faces with sad expressions, suggesting a tendency to linger on negative items as they were pushed off the mental list, possibly because these items activated related concepts and memories more easily. For instance, a sad face might trigger ruminative thoughts in a participant about a negative experience or their current dysphoria. Such an elaboration on mood-congruent items could also account for the observation that non-depressed control participants responded relatively slowly in trials that involved breaking a set of happy faces, as these items might trigger positive elaboration in healthy participants. Surprisingly, there was no evidence of a perceptual bias: in a 0-back control condition, in which participants compared each stimulus to a target expression given earlier, depressed and non-depressed participants performed identically.

We present a computational cognitive model that can explain the observed differences in performance between depressed participants and healthy controls, on the basis of simple assumptions about mental elaboration on mood-congruent items as they are removed from working memory, which can account for the behavioural differences between participants with and without depression. Our model also suggests that the existence of a perceptual bias is necessary for capturing depressed participants' faster responses to sad stimuli in the 2-back task.

Model

Following our previous model of rumination, performance on the emotional *n*-back task is modelled as a competition for cognitive resources between task-directed thought and off-task thought (i.e., mind-wandering or rumination).

The model is implemented in the PRIMs cognitive architecture (Taatgen, 2013), an extension of the ACT-R architecture (Anderson, 2007). Like ACT-R, PRIMs provides a framework for modelling an entire task, from perceptual input to motor output, and everything in between. The architecture consists of a number of modules (declarative memory, working memory, vision, goal state, motor action) that exchange information through buffers. A PRIMs model has one or more goals, each of which has a unique set of operators associated with it. PRIMs operators are comparable to production rules in ACT-R: if-then rules requiring certain conditions to be met before a sequence of information processing steps is performed. For example, a goal might be to memorise a letter shown on screen. An operator associated with this goal may first check that there is a letter, read the letter, and then encode it in a new memory chunk in working memory.

Unlike ACT-R, the PRIMs architecture makes no theoretical distinction between procedural knowledge (operators or productions) and declarative knowledge (facts): both are treated as regular chunks stored in declarative memory. This means that, like declarative facts, each operator has an activation value (representing its memory trace strength, which is boosted by retrievals and decays over time), and is subject to spreading activation. Goals spread activation to their operators, making operators belonging to more active goals more likely to be selected for execution. Operators can also spread activation to each other, thereby increasing the likelihood that operator execution follows a certain favoured sequence. Chunks in other buffers, such as the imaginal buffer or the retrieval buffer, can furthermore influence operator selection through spreading activation. Because of these dynamics, a PRIMs model can freely alternate operators from different goals. Our emotional *n*-back model switches flexibly between operators that fall under the umbrella of the task goal, and operators that enact the mind-wandering goal.

The model also builds on previous ACT-R models of the 2-back task. Lovett et al. (2000) and Juvina and Taatgen (2007) implemented multiple 2-back strategies. Here, we implement their high-effort strategy, since participants were explicitly instructed to use this method. This means that the model holds a list of the most recently seen stimuli in its declarative memory that is updated every trial. Each item on this list is a chunk, encoding the facial expression of the stimulus (*happy*, *neutral*, or *sad*) as well as its position in the list (*0-back*, *1-back*, or *2-back*). At the start of a trial, the model retrieves each item on the list in turn, increments its list position, and pushes the modified item back to declarative memory. In the case of the old 2-back chunk, its index is changed to *old*, reflecting the fact that it is no longer part of the list. The new stimulus that is currently on screen is then added to the front of the list. To decide its response, the model retrieves the 2-back item from memory and compares it to the current stimulus.

The recognition of facial expressions is modelled as a memory retrieval: the model sees a string representing a face, which it uses as a cue to retrieve the corresponding expression from its declarative memory. This mechanism allows us to model differences in recognition speed by varying the activation of face chunks of different emotional valence. Levens and Gotlib (2010) found that both depressed and non-depressed participants responded more quickly to happy faces than to sad or neutral faces, so chunks representing happy faces are given a higher activation, making them faster to retrieve. Since humans are not perfect at recognising facial expressions, we use partial matching to allow the occasional retrieval of an incorrect facial expression. Participants in both groups were more accurate in their responses to happy faces than to sad or neutral faces, with neutral faces being the most difficult to categorise (Levens & Gotlib, 2010). This effect is captured in the model by varying the mismatch penalty that is applied to a non-matching face on the basis of its expression.

The competing mind-wandering goal is implemented as a process of memory retrieval, following earlier models of mind-wandering (van Vugt, Taatgen, Sackur, & Bastian, 2015; van Vugt et al., 2018). At any point in the task, operators from this goal can initiate the retrieval of a sequence of random items from declarative memory. Once a memory is retrieved, it is placed in the model's working memory to bring it to the forefront of attention, and a new memory is retrieved. This process continues until a task operator is selected.

In addition to the 2-back condition, we also model the 0-back condition, which Levens and Gotlib (2010) used to show that depressed participants did not have a different perceptual bias than non-depressed controls. The 0-back model works the same as the 2-back model when it comes to perceiving and classifying stimuli, but differs in how it determines the correct response. In the 0-back condition, each stimulus has to be compared to a target expression, which eliminates the need for maintaining a history of recent stimuli in memory. Instead, once the model has determined the facial expression of a stimulus, it directly compares the expression to the target stored in its goal buffer, and responds accordingly.

To capture the differences between depressed and non-depressed participants, we manipulated several aspects of the model. Firstly, since depression is associated with more pervasive mind-wandering, the activation of mind-wandering operators was increased in the depressed version of the model, making them more likely to be selected. In addition, we recreated the general psychomotor slowing found in those with depression (Schrijvers, Hulstijn, & Sabbe, 2008) by increasing the time needed to execute a keypress in the depressed model. These two manipulations were expected to lower the depressed model's response rate on the *n*-back, while increasing its average response time.

Depression is linked to difficulties inhibiting negative emotional information that is no longer relevant, so the model includes a mechanism by which it can elaborate on sad faces as they are discarded. Whenever a sad face is pushed off the model's mental list of recent stimuli, the task process can be briefly interrupted by an elaboration operator from the mind-wandering goal. Although it is simply implemented as a delay in the current model, this operator represents the activation of related concepts and memories that are triggered by the sad face. The time penalty that is incurred by lingering on old information and subsequently having to refocus on updating the mental list will slow down the model's response in the affected trials. Since non-depressed individuals have been found to exhibit a difficulty letting go of positive emotional information, a similar mechanism is implemented in the non-depressed version of the model. Rather than elaborating on negative items, however, the non-depressed model has an operator that can elaborate on happy faces as they are removed from the mental list.

Finally, the depressed version of the 2-back model includes a negative perceptual bias: sad face chunks are given a higher activation than in the non-depressed version of the model, which makes them faster to retrieve. Since Levens and Gotlib (2010) did not find evidence of such a bias in the 0-back condition, it is left out of the 0-back model.

Parameters that distinguish the depressed model from the control model (activation of mind-wandering operators, keypress duration, and activation of face chunks) were initially set to reasonable values and were subsequently adjusted to better fit the behavioural results. The full code for depressed and control versions of the model is available at `github.com/maartenvandervelde/emotional-n-back/`.

Methods

0-back

The depressed and control version of the 0-back model were each run 50 times, simulating 50 participants per group (similar to the 29 participants per group in the empirical data). Both models went through 9 full training runs of 3 blocks with 43 trials each to learn the task, before they performed a final run. Only the final run of each model was analysed.

2-back

Both versions of the 2-back model were run 50 times, simulating 50 depressed participants and 50 control participants. The models received a similar amount of training as before (3 full runs containing 6 blocks of 55 trials), before performing a final run. Once again, only this last run was analysed.

Analysis

Performance of the depressed and control versions of the emotional *n*-back model was compared to that of the depressed and healthy participants in Levens and Gotlib (2010). Following Levens and Gotlib, we removed trials with response times outside 2.5 SD of a model participant's mean response time. Only correct trials were used for calculating mean response time. Trials in which the model did not respond within 2 seconds after stimulus onset were considered non-response trials. These trials were excluded when calculating response accuracy. To allow for further comparison of

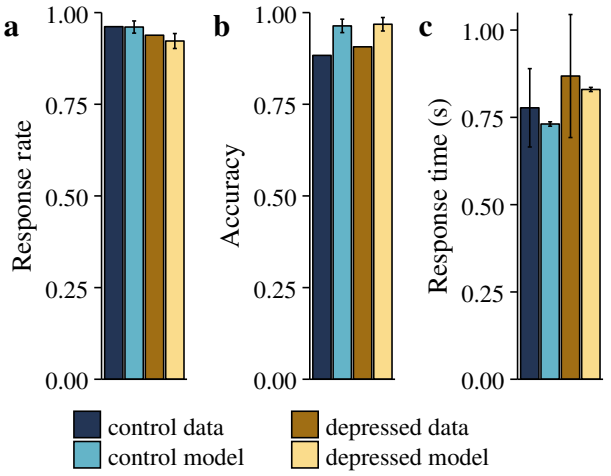


Figure 1: 0-back model performance compared to that of real participants. The model replicates the slightly lower response rate of depressed participants relative to controls (**a**), as well as the lack of a difference in accuracy between groups (**b**). As in humans, the depressed version of the model has longer response times (**c**). Error bars indicate 1 SD from the mean (SD of the human response rate and accuracy was unavailable).

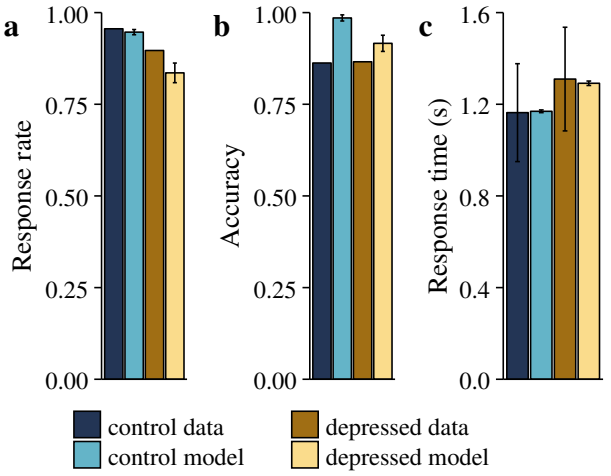


Figure 2: 2-back model performance compared to that of real participants. The model replicates the lower response rate of depressed participants relative to controls (**a**), but does not capture the similar response accuracy of both groups (**b**). As in humans, the depressed version of the model has longer response times (**c**). Error bars indicate 1 SD from the mean (SD of the human response rate and accuracy was unavailable).

relative response speed between groups, response times were normalised using a z-transformation (for every participant, each condition mean RT was subtracted from their overall mean RT and divided by the SD of the condition RT).

Results

0-back

Figure 1 summarises the 0-back model's behavioural fit to the human data[1]. It shows that, by increasing the model's tendency to engage in mind-wandering and by slowing down its motor response, we can reproduce the lower response rate and higher response times of depressed participants, relative to their non-depressed counterparts (although the model's response times are less variable than those of human participants). As in humans, depression does not affect response accuracy; the depressed and non-depressed versions of the model have equal accuracy (albeit slightly higher than the accuracy of human participants). Table 1 confirms that the model fits the data satisfactorily.

2-back

Figure 2 shows the fit of the 2-back model to the human data. As before, the higher activation of mind-wandering operators in the depressed version, together with a longer keypress time, causes it to correctly predict a lower response rate and higher response times relative to the control model. However, while participants in both groups responded equally accurately, the control model is more accurate than the depressed model. Both models' accuracy is slightly too high, but otherwise they capture the differences between groups (Table 1).

Z-transformed response times in two relevant 2-back conditions are shown in Figure 3. As Figure 3a shows, the model reproduces the relatively slow responses that occur in trials which require a mood-congruent face to be removed from the mental list. By elaborating on sad faces as they are being discarded, the depressed version of the model takes relatively long to respond in trials in which a previous set of sad faces is broken. In contrast, the control model is slower than usual in trials that require the breaking of an old set of happy faces, as it spends some time elaborating on the happy face being discarded. When breaking a neutral set, neither model elaborates on the discarded face. As a result, both are equally fast (accounting for differences in baseline speed), matching the

Table 1: Model fit to data. RMSE = root-mean-square error.

Task	Measure	RMSE
0-back	Response rate	.01
	Accuracy	.07
	Response time (s)	.04
2-back	Response rate	.04
	Accuracy	.09
	Response time (s)	.01
	Z-transformed RT	.46

[1]The output data of the model, as well as additional analyses of the results, including a more detailed breakdown by condition and statistical comparisons, are available at `github.com/maartenvandervelde/emotional-n-back/`.

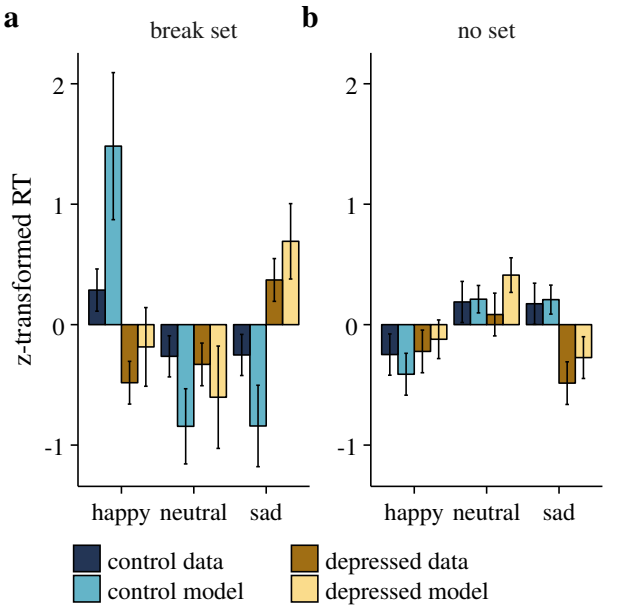


Figure 3: Z-transformed response times of the model compared to those of human participants on two conditions of the 2-back task. **(a)** When breaking a mental set (i.e., discarding one of the elements that previously made up a set from the mental list), the model reproduces the mood-congruent slowing effect seen in humans: the control model responds relatively slowly when breaking a happy set, while the depressed model is slow when breaking a sad set. Like humans, both models are equally fast when breaking a neutral set. (**b**) In non-match trials without a set to break, the depressed model responds more quickly to a sad stimulus than the control model, matching the pattern found in humans. Error bars indicate 1 SD from the mean.

pattern found in human participants.

Figure 3b further shows that, by manipulating the speed at which the model recognises a particular facial expression, we can recreate the human response time pattern observed in trials without a previous set to break or a current match. In particular, the higher activation of sad face chunks in the depressed model allows it to respond more quickly in trials with a sad stimulus, compared both to its own response times in happy or neutral trials and to the control model's responses in sad trials.

Discussion

In this paper we have presented a model of the cognitive effects of depression on performance in an emotional working memory task. Through the model, we have identified several key factors that differ between depressed and non-depressed participants. Depressed patients have a stronger tendency for mind-wandering, at the cost of attending to the task, which explains a lower response rate. Additionally, a slower motor response accounts for a higher average response time. Since individuals with depression were previously found to have difficulty inhibiting negatively-valenced information once it was no longer relevant, we implemented a mechanism for elaborating on negative items as they were discarded. With this mechanism, the model reproduces the observed response time pattern in the human data. The same mechanism, but focused on positive instead of negative items, furthermore captures the opposite pattern that was present in the non-depressed control group.

A potential criticism of our model is that it fails to provide a unified explanation for depressed participants' lack of a perceptual bias in the 0-back on one hand, and their faster responses to sad stimuli in the 2-back on the other. Depressed 0-back performance is captured without a perceptual bias relative to the control model. Yet, the depressed 2-back model requires faster perception of sad faces to reproduce depressed participants' relatively fast responses to sad faces.

There are two responses to this. Firstly, it is possible that the absence of perceptual bias in the 0-back task was a chance finding—indeed, it is inconsistent with earlier studies associating depression with a negative perceptual bias, and a study by the same authors in which participants who had recovered from depression performed the same task did find a sadness bias in the 0-back (Levens & Gotlib, 2015). If so, we could use the same response bias in the 0-back and 2-back models.

Alternatively, the inconsistency is due to an architectural limitation. It can be argued that the lower response times to sad stimuli in the 2-back task are not the result of faster perception, but rather of faster integration into working memory. Currently, PRIMs (and ACT-R) assumes that creating a new chunk in working memory takes a fixed length of time, irrespective of its contents. If it is indeed the case that depressed individuals integrate sad faces more quickly into working memory than neutral or happy faces, the architecture would have to be extended to support a variable working memory integration time that is dependent on an item's content.

An additional limitation of the model is that it assumes there to be little interaction between perceptual bias, working memory bias, and the mind-wandering process. Instead, these components are modelled as separate mechanisms. In an earlier version of the model we unsuccessfully used spreading activation in an attempt to influence task performance through the occurrence of mind-wandering. In that model, mind-wandering memories had an emotional valence (*happy*, *neutral*, or *sad*) and could spread activation to face chunks with their respective emotion, such that negatively-themed mind-wandering would make recognition of sad faces faster, and the processing of sad faces would more easily trigger mind-wandering. This approach failed because spreading activation is too fleeting, disappearing as soon as its source leaves its buffer. To make it work would require having a longer-lasting effect of spreading activation. Alternatively, affect-related spreading activation should come from an external source (such as physiology, see, e.g., ACT-R Φ (Dancy, 2013), which would also allow one to model some of the biologi-

cal factors underlying depression). A further option would be to model the tendency to elaborate on mood-congruent information as a learned association. Future work should examine whether these mechanisms could achieve a similar fit to the data while requiring fewer parameters.

In summary, we have shown how computational cognitive modelling can help us uncover the cognitive mechanisms that shape behaviour in depression and other psychiatric disorders. With a stronger propensity for mind-wandering, slower motor actions, a negative perceptual bias, and selective elaboration on mood-congruent items in working memory, our model reproduces specific behavioural deficits in working memory observed in humans. Through the implementation of existing theory, and by comparing the resulting predictions against human behaviour, we can deepen our understanding of the depressed mind.

References

Adams, R. A., Huys, Q. J. M., & Roiser, J. P. (2016). Computational Psychiatry: Towards a mathematically informed understanding of mental illness. *J Neurol Neurosurg Psychiatry*, *87*(1), 53–63.

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford, United Kingdom: OUP.

Bourke, C., Douglas, K., & Porter, R. (2010). Processing of facial emotion expression in major depression: A review. *Australian and New Zealand Journal of Psychiatry*, *44*(8), 681–696.

Christoff, K., Irving, Z. C., Fox, K. C. R., Spreng, R. N., & Andrews-Hanna, J. R. (2016). Mind-wandering as spontaneous thought: A dynamic framework. *Nature Reviews Neuroscience*, *17*(11), 718–731.

Dalgleish, T., & Watts, F. N. (1990). Biases of attention and memory in disorders of anxiety and depression. *Clinical Psychology Review*, *10*(5), 589–604.

Dancy, C. L. (2013). ACT-RΦ: A cognitive architecture with physiology and affect. *Biologically Inspired Cognitive Architectures*, *6*, 40–45.

Deveney, C., & Deldin, P. (2006). A preliminary investigation of cognitive flexibility for emotional information in major depressive disorder and nonpsychiatric controls. *Emotion*, *6*, 429–37.

Gollan, J. K., Pane, H. T., McCloskey, M. S., & Coccaro, E. F. (2008). Identifying differences in biased affective information processing in major depression. *Psychiatry Research*, *159*(1), 18–24.

Hoffmann, F., Banzhaf, C., Kanske, P., Bermpohl, F., & Singer, T. (2016). Where the depressed mind wanders: Self-generated thought patterns as assessed through experience sampling as a state marker of depression. *Journal of Affective Disorders*, *198*, 127–134.

Joormann, J., & Gotlib, I. H. (2008). Updating the contents of working memory in depression: Interference from irrelevant negative material. *Journal of Abnormal Psychology*, *117*(1), 182–192.

Juvina, I., & Taatgen, N. A. (2007). Modeling control strategies in the n-back task. In *Proceedings of the 8th International Conference on Cognitive Modeling* (pp. 73–78).

Levens, S. M., & Gotlib, I. H. (2010). Updating positive and negative stimuli in working memory in depression. *Journal of Experimental Psychology: General*, *139*(4), 654–664.

Levens, S. M., & Gotlib, I. H. (2015). Updating emotional content in recovered depressed individuals: Evaluating deficits in emotion processing following a depressive episode. *Journal of Behavior Therapy and Experimental Psychiatry*, *48*(Supplement C), 156–163.

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.

Marchetti, I., Koster, E. H., Klinger, E., & Alloy, L. B. (2016). Spontaneous Thought and Vulnerability to Mood Disorders The Dark Side of the Wandering Mind. *Clinical Psychological Science*, 835 – 857.

Mooneyham, B. W., & Schooler, J. W. (2013). The costs and benefits of mind-wandering: A review. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, *67*(1), 11–18.

Schrijvers, D., Hulstijn, W., & Sabbe, B. G. (2008). Psychomotor symptoms in depression: A diagnostic, pathophysiological and therapeutic tool. *Journal of Affective Disorders*, *109*(1-2), 1–20.

Smallwood, J., Obonsawin, M., & Heim, D. (2003). Task unrelated thought: The role of distributed processing. *Consciousness and Cognition*, *12*(2), 169–189.

Suslow, T., Junghanns, K., & Arolt, V. (2001). Detection of Facial Expressions of Emotions in Depression. *Perceptual and Motor Skills*, *92*(3), 857–868.

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471.

van Os, J., Gilvarry, C., Bale, R., Horn, E. V., Tattan, T., White, I., & Group, R. M. o. B. o. t. U. (1999). A comparison of the utility of dimensional and categorical representations of psychosis. *Psychological Medicine*, *29*(3), 595–606.

van Vugt, M. K., Taatgen, N. A., Sackur, J., & Bastian, M. (2015). Modeling mind-wandering: A tool to better understand distraction. In *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 252–257).

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.

Whitmer, A. J., & Banich, M. T. (2007). Inhibition versus switching deficits in different forms of rumination. *Psychological Science*, *18*(6), 546–553.

Whitmer, A. J., & Banich, M. T. (2010). Trait rumination and inhibitory deficits in long-term memory. *Cognition & Emotion*, *24*(1), 168–179.

ACTR-STAP: Connecting ACT-R to task software used by humans (and by other computational frameworks)

Vladislav D. Veksler (vdv718@gmail.com)
DCS Corp, U.S. Army Research Laboratory

Norbou Buchler
U.S. Army Research Laboratory

Abstract

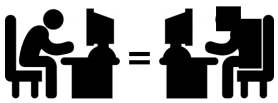
ACTR-STAP is an open-source ACT-R LISP library that enables ACT-R models to connect to task software developed in any programming language, running locally or remotely. Most importantly, the task software employed for ACT-R simulations would be the same software that is employed in human experiments and/or simulations with other modeling frameworks. STAP promotes model re-use across tasks and task re-use across models (including non ACT-R models). ACTR-STAP enables millisecond-precision model-controlled timing, even for faster-than-real-time simulations, and STAP message log files may be played back and analyzed on equal footing with those of humans and other computational participants.

Keywords: STAP, simulations, cognitive modeling

Research in computational cognitive modeling often involves the comparison of model behavior to that of humans, and the comparison of multiple models to each other across a set of tasks. Unfortunately, graphic displays as seen by human participants (Graphical User Interfaces; GUIs) do not naturally translate to virtual display components required by cognitive modeling frameworks. Likewise, a simulated task developed for use with a cognitive modeling framework would not inherently include a GUI for human participants. Moreover, a task development effort that specifically focuses on adding user interface components into both GUI and computational framework displays often focuses on a single type of cognitive framework, making it difficult to do cross-framework model comparison. A comparison of human behavior to that of several models based in different computational cognitive architectures would require a multiplication of the development effort.

Though this may seem like a menial software-development issue, it bears significant consequences for scientific progress. A given empirical study may benefit from a complementary computational simulation, but the benefits often do not merit the effort needed to try and connect a cognitive framework to the software used in the human study. In another case, a new computational model of cognitive processes may benefit from evaluation across a wide variety of task software, but the development effort needed to parse each task interface (or to re-develop each task for the purposes of having access to their interfaces) is prohibitive. The development effort needed to gather data from human participants and multiple models across several cognitive frameworks across multiple tasks of any complexity is simply beyond what any research laboratory can afford. Lowering the bar to entry for connecting computational agent frameworks to a battery of behavioral software – the same software that is employed in human studies – would enable a better quality of research.

The Simple Task-Actor Protocol (STAP; https://vdv7.github.io/stap; Veksler, Buchler, Lebiere, & Morrison, 2018; Veksler, Buchler, Lebiere, Morrison, & Kelley, 2016) focuses on *universal task access* – enabling equal access to human and computational participants, regardless of user-side device, operating system, or programming paradigm. More specifically, STAP is a user interface (UI) serialization language for enabling plug-and-play interconnectivity between software employed in human studies and various modeling frameworks and computational agents. Some of the main features of STAP include (1) clear separation of functionally-essential task affordances from arbitrary stylistic choices, (2) minimal core set of UI primitives, and (3) user-side time management (for faster-than-real-time and slower-than-real-time simulations). Additionally, STAP messages are JSON (http://json.org) format compliant, which means that STAP messages can be serialized/deserialized via JSON libraries available for just about every programming language, including LISP, Python, and JAVA (i.e. the languages most-often employed for ACT-R modeling).



The ACT-R cognitive architecture (Anderson, 1993, 2007; Anderson et al., 2004; Anderson & Lebiere, 1998) is a popular framework for developing models of cognition and running behavioral simulations. This is a very important framework, not only as a theory of cognition, but also as software that comprises empirically validated cognitive mechanisms capable of predicting timing and preferences at the level of milliseconds, as well as practice/performance effects at the level of days, months, and years. Perhaps just as important as the software capabilities themselves is that the domain-specific language employed for ACT-R modeling is widely known and employed by hundreds of Cognitive Science researchers across the world.

Most often ACT-R models and tasks are developed in LISP, although there are ACT-R implementations in JAVA and in Python programming languages, as well. It is rare, however,

for ACT-R models developed in one programming paradigm to be employed for simulations where task software is developed in another one. An HTML5 task developed for mass human participation via Mechanical Turk, for example, will not be accessible by an ACT-R model developed in LISP. A standard ACT-R modeling effort involves (1) the development of experiment software and a model within the same environment (e.g. LispWorks), with custom calls to functions that add UI primitives to both, the human GUI, and the ACT-R visicon (ACT-R internal virtual display); (2) the development of GUI experiment software for humans, and a separate replication of this effort for ACT-R simulations; or (3) in the case where the software employed for human experiments provides an API (Application Programming Interface), developing a custom translation layer for this API for ACT-R.

The lack of a standard method/API for connecting ACT-R models to task software greatly increases the development effort needed for each simulation, essentially reducing the set of potential tasks that models may be connected to. There are two additional problems. First, the virtual display in ACT-R, whether based on an exposed task API or based on a reconstructed simulation of the task, is not guaranteed to include every important detail of the GUI task software that humans are exposed to, reducing the validity of human-model comparisons. Second, given the effort needed to replicate the task or develop an API translation layer for ACT-R, it is unlikely that the researcher will repeat this effort for connecting a different cognitive framework to the task, making cross-framework model comparison (however much wanted) highly unlikely. A standard API, such as STAP, would help to resolve these problems, promoting model re-use across tasks, and task-reuse across models and modeling frameworks.

ACTR-STAP (https://github.com/vdv7/actr-stap) is an open-source LISP library developed for connecting ACT-R via TCP to any STAP-compliant task software. STAP task software may be developed in any programming language, and may be presented to human participants on any device. Thus, ACTR-STAP enables modelers to run ACT-R simulations using the same task software that is employed by human participants, whether the software was developed for controlled laboratory studies or for open web access. Unlike prior efforts to connect ACT-R to task software via a standard API (e.g., Hope, Schoelles, & Gray, 2014), STAP isn't specific to either ACT-R, nor cognitive-modeling. STAP simply enables and promotes the separation of functionally-relevant task affordances from arbitrary stylistic choices in the UI, thus making it easier for machines to parse the information and respond. In this way, STAP, as a language, is highly compatible with the ACT-R visicon (as it would be with virtual displays in other cognitive and AI frameworks).

More specifically, each STAP message from task software is an update to the display. ACTR-STAP opens a connection to task software (which runs as a server), and then pushes updates to the ACT-R visicon based on the incoming STAP messages (i.e., JSON arrays comprising added/updated UI element properties). Any ACT-R device inputs (e.g., button clicks) are asynchronously sent back to the task server as STAP-compliant response (i.e., JSON arrays comprising the input element and value, along with the ACT-R time stamp).

Event timing in STAP is driven by user-side software, which enables millisecond-level sensitivity in modeling research, even in faster-than-real-time simulations. Additionally, user-side time-stamps enable real-time playback of non-real-time simulations. Since logged STAP messages look the same regardless of whether the participant is human, or an ACT-R model, or some other computational agent, the logs from different participant types may be analyzed on equal footing (or played back for Turing testing).

STAP-compliant task software is unambiguous as to which UI features are required for task participation (these are specified via a "require" directive). If a model was to be exposed to a battery of tasks, and one of the tasks required a feature that is yet to be implemented in ACTR-STAP (e.g. animation), the model could simply skip this task. ACTR-STAP currently enables all text-and-button tasks, as well as precision timing delay/wait options. Importantly, more advanced feature development efforts would not be one-off efforts unusable beyond specific experiments. Rather, as the community continues to drive this open-source library, ACT-R modelers will be able to connect to a continuously increasing range of tasks.

References

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

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

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

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), 1007–12. doi: 10.3758/s13428-013-0425-z

Veksler, V. D., Buchler, N., Lebiere, C., & Morrison, D. (2018). Humans aren't enough: Providing access for simulated participants to behavioral experiment software. In *40th annual meeting of the cognitive science society (cogsci 2018)*.

Veksler, V. D., Buchler, N., Lebiere, C., Morrison, D., & Kelley, T. D. (2016). The performance comparison problem: Universal task access for cross-framework evaluation, Turing tests, grand challenges, and cognitive decathlons. *Biologically Inspired Cognitive Architectures*(18C), 9–22. Retrieved from `http://dx.doi.org/10.1016/j.bica.2016.10.003`

ACTR-STAP: Connecting ACT-R to task software used by humans (and by other computational frameworks)

Vladislav D. Veksler (vdv718@gmail.com)
DCS Corp, U.S. Army Research Laboratory

Norbou Buchler
U.S. Army Research Laboratory

Abstract

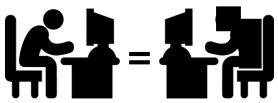
ACTR-STAP is an open-source ACT-R LISP library that enables ACT-R models to connect to task software developed in any programming language, running locally or remotely. Most importantly, the task software employed for ACT-R simulations would be the same software that is employed in human experiments and/or simulations with other modeling frameworks. STAP promotes model re-use across tasks and task re-use across models (including non ACT-R models). ACTR-STAP enables millisecond-precision model-controlled timing, even for faster-than-real-time simulations, and STAP message log files may be played back and analyzed on equal footing with those of humans and other computational participants.

Keywords: STAP, simulations, cognitive modeling

Research in computational cognitive modeling often involves the comparison of model behavior to that of humans, and the comparison of multiple models to each other across a set of tasks. Unfortunately, graphic displays as seen by human participants (Graphical User Interfaces; GUIs) do not naturally translate to virtual display components required by cognitive modeling frameworks. Likewise, a simulated task developed for use with a cognitive modeling framework would not inherently include a GUI for human participants. Moreover, a task development effort that specifically focuses on adding user interface components into both GUI and computational framework displays often focuses on a single type of cognitive framework, making it difficult to do cross-framework model comparison. A comparison of human behavior to that of several models based in different computational cognitive architectures would require a multiplication of the development effort.

Though this may seem like a menial software-development issue, it bears significant consequences for scientific progress. A given empirical study may benefit from a complementary computational simulation, but the benefits often do not merit the effort needed to try and connect a cognitive framework to the software used in the human study. In another case, a new computational model of cognitive processes may benefit from evaluation across a wide variety of task software, but the development effort needed to parse each task interface (or to re-develop each task for the purposes of having access to their interfaces) is prohibitive. The development effort needed to gather data from human participants and multiple models across several cognitive frameworks across multiple tasks of any complexity is simply beyond what any research laboratory can afford. Lowering the bar to entry for connecting computational agent frameworks to a battery of behavioral software – the same software that is employed in human studies – would enable a better quality of research.

The Simple Task-Actor Protocol (STAP; https://vdv7.github.io/stap; Veksler, Buchler, Lebiere, & Morrison, 2018; Veksler, Buchler, Lebiere, Morrison, & Kelley, 2016) focuses on *universal task access* – enabling equal access to human and computational participants, regardless of user-side device, operating system, or programming paradigm. More specifically, STAP is a user interface (UI) serialization language for enabling plug-and-play interconnectivity between software employed in human studies and various modeling frameworks and computational agents. Some of the main features of STAP include (1) clear separation of functionally-essential task affordances from arbitrary stylistic choices, (2) minimal core set of UI primitives, and (3) user-side time management (for faster-than-real-time and slower-than-real-time simulations). Additionally, STAP messages are JSON (http://json.org) format compliant, which means that STAP messages can be serialized/deserialized via JSON libraries available for just about every programming language, including LISP, Python, and JAVA (i.e. the languages most-often employed for ACT-R modeling).



The ACT-R cognitive architecture (Anderson, 1993, 2007; Anderson et al., 2004; Anderson & Lebiere, 1998) is a popular framework for developing models of cognition and running behavioral simulations. This is a very important framework, not only as a theory of cognition, but also as software that comprises empirically validated cognitive mechanisms capable of predicting timing and preferences at the level of milliseconds, as well as practice/performance effects at the level of days, months, and years. Perhaps just as important as the software capabilities themselves is that the domain-specific language employed for ACT-R modeling is widely known and employed by hundreds of Cognitive Science researchers across the world.

Most often ACT-R models and tasks are developed in LISP, although there are ACT-R implementations in JAVA and in Python programming languages, as well. It is rare, however,

for ACT-R models developed in one programming paradigm to be employed for simulations where task software is developed in another one. An HTML5 task developed for mass human participation via Mechanical Turk, for example, will not be accessible by an ACT-R model developed in LISP. A standard ACT-R modeling effort involves (1) the development of experiment software and a model within the same environment (e.g. LispWorks), with custom calls to functions that add UI primitives to both, the human GUI, and the ACT-R visicon (ACT-R internal virtual display); (2) the development of GUI experiment software for humans, and a separate replication of this effort for ACT-R simulations; or (3) in the case where the software employed for human experiments provides an API (Application Programming Interface), developing a custom translation layer for this API for ACT-R.

The lack of a standard method/API for connecting ACT-R models to task software greatly increases the development effort needed for each simulation, essentially reducing the set of potential tasks that models may be connected to. There are two additional problems. First, the virtual display in ACT-R, whether based on an exposed task API or based on a reconstructed simulation of the task, is not guaranteed to include every important detail of the GUI task software that humans are exposed to, reducing the validity of human-model comparisons. Second, given the effort needed to replicate the task or develop an API translation layer for ACT-R, it is unlikely that the researcher will repeat this effort for connecting a different cognitive framework to the task, making cross-framework model comparison (however much wanted) highly unlikely. A standard API, such as STAP, would help to resolve these problems, promoting model re-use across tasks, and task-reuse across models and modeling frameworks.

ACTR-STAP (https://github.com/vdv7/actr-stap) is an open-source LISP library developed for connecting ACT-R via TCP to any STAP-compliant task software. STAP task software may be developed in any programming language, and may be presented to human participants on any device. Thus, ACTR-STAP enables modelers to run ACT-R simulations using the same task software that is employed by human participants, whether the software was developed for controlled laboratory studies or for open web access. Unlike prior efforts to connect ACT-R to task software via a standard API (e.g., Hope, Schoelles, & Gray, 2014), STAP isn't specific to either ACT-R, nor cognitive-modeling. STAP simply enables and promotes the separation of functionally-relevant task affordances from arbitrary stylistic choices in the UI, thus making it easier for machines to parse the information and respond. In this way, STAP, as a language, is highly compatible with the ACT-R visicon (as it would be with virtual displays in other cognitive and AI frameworks).

More specifically, each STAP message from task software is an update to the display. ACTR-STAP opens a connection to task software (which runs as a server), and then pushes updates to the ACT-R visicon based on the incoming STAP messages (i.e., JSON arrays comprising added/updated UI element properties). Any ACT-R device inputs (e.g., button clicks) are asynchronously sent back to the task server as STAP-compliant response (i.e., JSON arrays comprising the input element and value, along with the ACT-R time stamp).

Event timing in STAP is driven by user-side software, which enables millisecond-level sensitivity in modeling research, even in faster-than-real-time simulations. Additionally, user-side time-stamps enable real-time playback of non-real-time simulations. Since logged STAP messages look the same regardless of whether the participant is human, or an ACT-R model, or some other computational agent, the logs from different participant types may be analyzed on equal footing (or played back for Turing testing).

STAP-compliant task software is unambiguous as to which UI features are required for task participation (these are specified via a "require" directive). If a model was to be exposed to a battery of tasks, and one of the tasks required a feature that is yet to be implemented in ACTR-STAP (e.g. animation), the model could simply skip this task. ACTR-STAP currently enables all text-and-button tasks, as well as precision timing delay/wait options. Importantly, more advanced feature development efforts would not be one-off efforts unusable beyond specific experiments. Rather, as the community continues to drive this open-source library, ACT-R modelers will be able to connect to a continuously increasing range of tasks.

References

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

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

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

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), 1007–12. doi: 10.3758/s13428-013-0425-z

Veksler, V. D., Buchler, N., Lebiere, C., & Morrison, D. (2018). Humans aren't enough: Providing access for simulated participants to behavioral experiment software. In *40th annual meeting of the cognitive science society (cogsci 2018)*.

Veksler, V. D., Buchler, N., Lebiere, C., Morrison, D., & Kelley, T. D. (2016). The performance comparison problem: Universal task access for cross-framework evaluation, Turing tests, grand challenges, and cognitive decathlons. *Biologically Inspired Cognitive Architectures*(18C), 9–22. Retrieved from `http://dx.doi.org/10.1016/j.bica.2016.10.003`

The time course recovery of confidence judgments using interruptions

Kevin Zish (kzish@gmu.edu)
George Mason University
4400 University Dr, Fairfax, VA 22030

Nathan Aguiar (naguiar@masonlive.gmu.edu)
George Mason University
4400 University Dr, Fairfax, VA 22030

Malcolm McCurry (malcolm.mccurry.ctr@nrl.navy.mil)
Peraton
12975 Worldgate Dr, Herndon, VA 20170

Erik M. Altmann (ema@msu.edu)
Michigan State University
316 Physics Rd, East Lansing, MI 48824

J. Gregory Trafton (greg.trafton@nrl.navy.mil)
U.S. Naval Research Laboratory
4555 Overlook Ave SW, Washington, DC 20375

Abstract

Participants were interrupted during a procedural ask. Choice response time and confidence were measured up to seven trials after the interruption. Empirical data suggests a curvilinear pattern of recovery for choice response time and confidence. Two models of recovery for choice response time and confidence judgments were built. A comparison of the models provided support for post-decisional theories of confidence which suggest that confidence judgments are formed after a choice is made.

Keywords: ACT-R, confidence, activation, modeling, decision-making

Introduction

When are confidence judgments formed? Current theories of confidence debate whether confidence judgments are formed at the time a decision is made or after. If someone asks: "How confident are you that the person you identified committed the crime?", decisional theories of confidence say that confidence is available at the same time as the decision. Post-decisional theories of confidence say that confidence is only available after the decision is made.

Decisional theories are modeled in the context of signal detection theory (SDT: Green & Swets, 1966) and the role of strength in recognition memory (Egan, Schulman, & Greenberg, 1959; Hart, 1967; Norman & Wickelgren, 1969; Wickelgren, 1968). In these theories, confidence judgments are directly related to the strength of a retrieved memory. Stronger memories elicit higher confidence responses than weaker memories. Confidence is the strength of a memory or the memory's distance to a decision criterion (Donaldson, 1996; Wixted & Mickes, 2010). The confidence judgment is tied to the decision process and confidence is available at the same time a decision is made.

Post-decisional theories state that forming a confidence judgment begins after a decision is made (Pleskac & Busemeyer, 2010; Vickers, 2001, 2014). These theories typically use sequential sampling models (Juslin & Olsson, 1997; Vickers, 1970), specifically drift diffusion models (Heath, 1984; Laming, 1968; Link & Heath, 1975; Ratcliff, 1978). In drift diffusion models, choice begins at some point z and evidence accrues on a series of counters (usually one) towards a criterion for decision A or decision B. Confidence is calculated as the speed with which evidence accumulates towards a criterion (Pleskac & Busemeyer, 2010; Ratcliff & Starns, 2009, 2013) or as the difference between the counters (Merkle & Van Zandt, 2006; Van Zandt & Maldonado-Molina, 2004; Vickers, 2014).

There is evidence for both decisional and post-decisional theories of confidence. In a recent study, Dotan, Meyniel, & Dehaene (2018) provided support for decisional theories of confidence. Participants were presented with arrows that pointed to the left or right side of the screen. Arrows were presented one at a time. Participants were asked to move their finger from the bottom middle to targets at either the top left or top right of the screen. Participants moved to the side they thought the majority of the arrows was pointing towards. The authors found that y-speed, the speed that a participant was heading towards one choice or the other, was a better predictor of confidence than choice response time (RT). Because instantaneous speed was a better predictor of confidence than final choice RT, the authors argued that confidence is calculated online and repeatedly throughout the process of a judgment.

Post-decisional theories suggest that evidence continues to be collected after a choice is made as shown by response

reversals (Resulaj, Kiani, Wolpert, & Shadlen, 2009; Van Zandt & Maldonado-Molina, 2004). Therefore, confidence judgments are based on the evidence collected for the choice plus some additional evidence collection.

One influential post-decisional theory is the two stage dynamic signal detection theory (2DSD: Pleskac & Busemeyer, 2010) which models the relationship between choice and confidence in a drift diffusion model. Participants collect evidence for a choice using a standard drift diffusion process. Evidence retrieved from memory or the environment accrues on a counter towards a response threshold for given alternatives. When the counter reaches the choice criteria a choice is made. The evidence collected for the choice continues to be sampled to make a confidence judgment.

There is also support for both decisional and post-decional explanations in a single experiment (Baranski & Petrusic, 1998). Baranski & Petrusic (1998) asked participants to determine the longer or shorter of two horizontal lines under accuracy stress or speed stress. Under speed stress, the authors found a negatively linear relationship for confidence response time and confidence, where faster responses were more confident than slower responses. Under accuracy stress there was no relationship. The authors argued that speed stress caused participants to spend less time calculating confidence during the primary decision and rendered confidence post-decisonally. In contrast, participants had no change in confidence response time across levels of confidence for accuracy stress because confidence was processed during the decision.

The evidence from Dotan et al. (2018), 2DSD, and Baranski & Petrusic (1998) suggest that rendering confidence judgments is a cognitively complex process that can originate at different times during decision making.

One way to investigate cognitively complex processes is to interrupt them and record performance data at different points in time as the process recovers. Altmann & Trafton (2007) used interruptions to investigate time course recovery of memory and attention after an interruption.

Here, we incorporated confidence judgments in a procedural task that generates rich error data. Error data allows us to investigate process recovery of confidence judgments. Confidence judgments are relevant to study in this context because they could help inform both models of information processing in procedural tasks and performance in applied contexts where procedural errors have high potential cost.

Altmann & Trafton (2007) had participants complete a dynamic decision making task. Participants were interrupted after completing an action. The time to resume the task after the interruption (i.e. resumption lag: Trafton, Altmann, Brock, & Mintz, 2003) was measured up to seven actions after the interruption was complete. The authors found that interruptions increased resumption time after the interruption. However, participants did not immediately recover after the first action following an interruption. Instead, the data showed that resumption lag followed a curvilinear pattern of recovery wherein the response time for each action after the interruption was faster than the preceding action.

Altmann & Trafton (2007) developed a mathematical model that fit the pattern of recovery for resumption time. The model suggests that in procedural tasks spreading activation plays a critical role in facilitating the selection of the next action in a task. Activation spreads through associative links that form between actions in a task. An associative link means that when an action is completed, priming from the completed action is added to the activation of all following actions. The activation for an element at position *p* is represented by

$$A(p) = -1 + \sum_{i=1}^{p} assoc^{i-1}, assoc < 1$$

where *assoc* is the amount of activation received by preceding elements of the task that have already been retrieved. The action directly after the current action receives the most priming. Subsequent actions receive lower and lower amounts of priming.

When participants are allowed to complete an uninterrupted task, an element receives small amounts of priming from each associatively linked action before it. Interruptions effectively cut off that priming making it so that activation for the action to be resumed is lower than if it had not been interrupted. The curvilinear pattern of choice RT is evidence of cumulative priming building for the task as it recovers.

The recovery process of choice RT is based on the ACT-R theory (Anderson et al., 2004) and is represented by

$$RT(p) = F * exp[1 - \sum_{i=1}^{p} assoc^{i-1}]$$

where *F* is a scaling parameter representing non-decisional processes.

Using a similar approach, we can investigate the time course of confidence judgments by interrupting participants during a decision and building a model of choice, taken from Altmann & Trafton (2007), and a model of confidence as those processes recover. We assume, as does SDT theory, that confidence is a scaled measure of strength. We represent strength using activation which is the *assoc* parameter in the Altmann & Trafton (2007) model.

Activation, has been explored extensively in the ACT-R cognitive architecture (Anderson et al., 2004) of which the findings of Altmann & Trafton (2007) were based. We used the activation-based properties of ACT-R to make predictions about the relationship between choice RT and confidence.

ACT-R suggests that activation of a memory element *m* is defined by the relationship between the number of times a goal has been rehearsed *n* and the time that has passed *T*. The following is a simplified equation for activation adapted from equation 2.2 in Anderson, Bothell, Lebiere, & Matessa (1998).

$$m = ln(n/\sqrt{T})$$

Goals that have been rehearsed many times in the recent past have more activation than goals that have been rehearsed fewer times or rehearsed in the distant past.

Goals also undergo decay. Decay is an important part of forgetting and is indexed by time. The more time that has passed since a goal has received activation (from being retrieved or from associative priming) the lower the activation for the goal. Interruptions decrease activation by decreasing the probability that a goal can be rehearsed, increasing the amount of time between rehearsal, or some combination of the two.

Because we assume that confidence is based on activation, it follows that confidence should behave in several systematic ways according to ACT-R. First, confidence should decrease after an interruption because interruptions decrease activation. This finding has already been demonstrated by Aguiar, Zish, McCurry, & Trafton (2016) and Zish, Hassanzadeh, McCurry, & Trafton (2015).

Second, confidence should increase in a curvilinear pattern after an interruption similar to Altmann & Trafton (2007). Cumulative priming after an interruption should result in a decrease in confidence after an interruption followed by a gradual increase in confidence for later actions as the decision process recovers.

Third, a mathematical model of the time course of recovery for confidence *C* can be built that will match empirical data. The model we propose is

$$C(p) = S * exp[-1 + \sum_{i=1}^{p} assoc^{i-1}]$$

Different parameters were used for the RT model and the confidence model as RT and confidence are not on the same scale. We change the *F* parameter to *S* and the activation parameter from *–A(p)* for RT to *A(p)* given that RT decreases after an interruption and confidence increases.

Fourth, we can use the two models to provide evidence for decisional or post-decisional theories. In particular we can compare the *assoc* parameter in each model (RT and confidence). Decisional theories would predict that the *assoc* parameter for the RT model and the *assoc* parameter for the confidence model should be equivalent because the theories claim that both choice and confidence emerge at the same time. In contrast, post-decisional theories would predict that choice and confidence emerge at different times and in particular that confidence judgments occur after decision. Therefore, post-decisional theories would predict that the *assoc* parameter would be significantly smaller for the confidence model than the *assoc* parameter for the RT model.

Methods

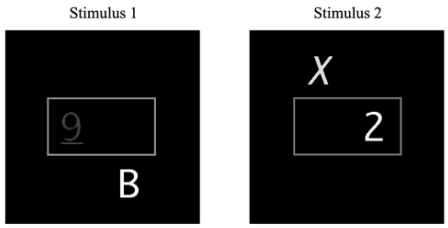
Participants

One hundred and fifty-five George Mason University undergraduates participated for course credit.

Task

Primary Task The UNRAVEL task was adapted from Altmann, Trafton, & Hambrick (2014). The UNRAVEL task has seven rules each represented by a letter (i.e. U, N, R, A, V, E, L). Participants are presented with one number and one letter at the same time. Each letter and number has certain characteristics that change from trial to trial such as color, font, position, etc. Participants are instructed to keep the UNRAVEL rule in memory, interpret what characteristic of the stimuli they are asked to identify, analyze the stimuli, and using the keyboard to submit what characteristic they identified (Figure 1). The UNRAVEL rules are available to the participant at anytime by holding the Shift + ? keys.

(a) Sample stimuli for UNRAVEL task:



(b) Choice rules and candidate responses for UNRAVEL task, and responses to stimuli in (a):

Step	Candidate responses		Choice rules	Responses to sample stimuli: Stimulus 1	Stimulus 2
U	u	i	character is Underlined or in Italics	u	i
N	n	f	letter is Near to or Far from start of alphabet	n	f
R	r	y	character is Red or Yellow	r	y
A	a	b	character is Above or Below the box	b	a
V	v	c	letter is Vowel or Consonant	c	c
E	e	o	digit is Even or Odd	o	e
L	l	m	digit is Less than or More than 5	m	l

Figure 1. Example of the UNRAVEL task from Altmann et al. (2014).

For example, the U action in UNRAVEL prompts participants to identify if a number or letter is underlined or italicized. If a letter or number is underlined they press the "u" key on the board. If the letter is italicized they are instructed to press the "i". After they submit their response participants will be presented with a brand new stimulus. They will search the stimulus for a characteristic prompted by the N action. The N action prompts participants to determine if the letter is near ("n") or far ("f") from the beginning of the alphabet. Participants continue to proceed through the UNRAVEL rules. Once completed, participants wrap around to the U action. The goal is to complete the rules in order and correctly identify the prompted characteristic for each stimulus.

Each action in UNRAVEL has a different set of keys associated with a response. As a result, the keystrokes reveal what action participants think they are on allowing for an analysis of sequence errors.

Interruption Task We used an equation to determine when participants were given a secondary task that served as an interruption. The equation can be found in Altmann et al., (2014) and resulted in an interruption 11.85% of the time.

After an UNRAVEL response was submitted, the UNRAVEL task was occluded and participants were asked to type in a series of letters into a box. Once the letters were typed in correctly the UNRAVEL task was revealed again. Participants were asked to return to the UNRAVEL task in the correct order.

Confidence Question Participants received a confidence question after completing an UNRAVEL action following half of the interruptions and an equal number of the control trials. The screen was replaced with a question that asked: "How confident were you that you just chose the correct step during the UNRAVEL task? Enter your choice on a scale from 1 to 6, with 1 being least confident and 6 being most confident." The participant typed in their response into a text field. After submitting their response the participant was returned to the UNRAVEL task

Procedure

Participants filled out an approved IRB consent form as well as biographical information. The task was first described using screenshots.

Participants were given a practice session where each rule of UNRAVEL was explained. They were exposed to all elements of the task including interruptions and confidence questions. Participants were shown that they could hit a certain key to access a list of the UNRAVEL rules at any time.

Results

One hundred and fifty-five participants completed 42442 UNRAVEL actions. We treated each action as a trial. There were 4925 confidence judgments. Only trials with confidence judgments were analyzed. We averaged RT and confidence for each participant for the first seven actions after an interruption.

Modeling the Time course of Recovery for Decisions RT

Not every participant had a confidence question at each step after the interruption. We used a linear mixed-effects model which can account for unbalanced repeated measures designs (Lindstrom & Bates, 1990) to look for differences in RT across step. There was a significant effect of step [$F(1, 723.61) = 69.16$, $p < .05$]. To investigate differences between steps, we compared Step 1 with Steps 2-7. Step1 after an interruption was significantly higher than Steps 2-7 [$F(1,154) = 171.9$, MSE = 207.15, $p < .05$]. This result replicates the disruptive effects of interruptions (Altmann et al., 2014; B. Edwards & Gronlund, 1998; Gillie & Broadbent, 1989).

Replicating Altmann & Trafton (2007) the response time for the primary judgment has a curvilinear pattern of recovery after an interruption [Overall: $F(6, 848) = 17.64$, $p < .05$; Linear: $t = -6.89$, $p < .05$; Quadratic: $t = 4.65$, $p < .05$]. Following Altmann & Trafton (2007), we fit the model to the data by estimating *F* and *assoc* for each participant. We used the mean decision RT for the first trial after the interruption for each participant as the *F* parameter. An RMSE was calculated for the *F* parameter while varying the *assoc* parameter between .005 and 1 by .005. The lowest RMSE for each fixed *F* and varied *assoc* parameter was set for each participant. The *F* and *assoc* were then averaged across participants to give us an *F* of 4.79 and an *assoc* of .29 for our model. The mean RMSE was 1.11 and R^2 was .37. Figure 2 shows the empirical data for choice RT for each UNRAVEL step after an interruption and predicted choice RT from our model. This replicates Altmann & Trafton (2007).

To test goodness of fit we ran runs tests (Bradley, 1968) on the signs of the deviations from the model minus the data. The runs test showed that the model and data were not significantly different from each other [$t(154) = 1.47$, $p = .14$].

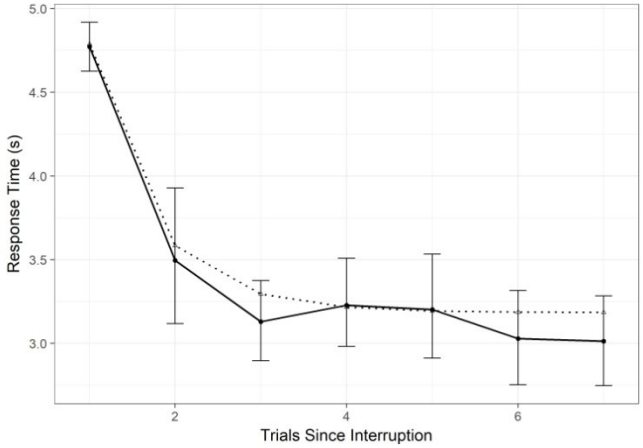


Figure 2. Data (solid) and model (dashed) for time course recovery of decision RT. Error bars are 95% confidence intervals data.

Modeling the Time course of Recovery for Confidence

There was a significant effect of step [$F(1, 701.85 = 35.80$, $p < .05$]. To investigate the differences between steps we compared Step 1 with Steps 2-7. Step 1 after an interruption was significantly lower than Steps 2-7 [$F(1,154) = 67.6$, MSE = 11.32, $p < .05$]. Similar to decision RT, confidence shows a pattern of recovery after an interruption [Overall: $F(6,848) = 3.89$, $p < .05$; Linear: $t = 3.05$, $p < .05$; Quadratic: $t = -2.37$, $p < .05$]. We used the same modeling process that Altmann & Trafton (2007) did and that was used above for the RT model. The *S* and *assoc* parameters were estimated for each participant and the parameters with the lowest RMSE were used for the final model. The *S* parameter was 4.95 and the *assoc* parameter was .08 for the final model. The mean RMSE was .41 and R^2 was .43. Figure 3 shows the empirical data for confidence for each UNRAVEL action after an interruption and predicted confidence from our model.

The goodness of fit for the runs test for the model and data showed no significant difference [t(154)= -.02, p = .98] suggesting that confidence is, indeed, tied to activation.

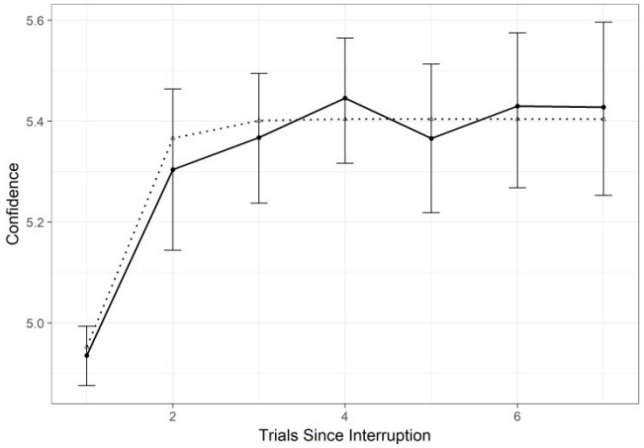


Figure 3. Data (solid) and model (dashed) for time course recovery of confidence. Error bars are 95% confidence intervals for data.

Comparing Activation for Choice RT and Confidence

To compare the timing of choice and confidence, we assume that the strength of a memory (activation) is the driver of both judgments.

Both of our models have an activation component that is represented by the *assoc* parameter. We took the *assoc* parameter from each participant's lowest RMSE model fit and compared *assoc* for choice RT and confidence using a within-subjects ANOVA. The *assoc* parameter was significantly lower for confidence (M=.08) than for choice RT (M=.29) [$F(1,154) = 169.9$, $MSE = 3.56$, $p < .05$, $\eta 2 = .31$] .

Discussion

In this paper we built two models of complex cognitive processes: decision-making and confidence judgments. We instantiated decision-making using the model from Altmann & Trafton (2007) and modeling choice RT. We then built a model of confidence that also used an activation parameter so that we could compare the models. Our data and models produced five important findings.

First, we were able to replicate the empirical curvilinear pattern from Altmann & Trafton (2007) and show that RT recovers over time after an interruption.

Second, we built and replicated a model of RT for data on a new task.

Third, we showed that confidence is influenced by cumulative priming and that confidence recovers over time. This is a unique finding given that many models and experiments measuring confidence consider the contribution of memory only from the item just retrieved (DeSoto & Roediger, 2014; Merkle & Van Zandt, 2006; Pleskac & Busemeyer, 2010; Ratcliff & Starns, 2009, 2013; Van Zandt & Maldonado-Molina, 2004; Douglas Vickers, 2014).

In this study we did not rely on the common list-learning or perceptual stimuli that have come to dominate the field. Instead we used a procedural task which allowed us to demonstrate that confidence judgments respond to priming from other elements of the task. Given that many of the current popular models of confidence consider confidence judgments a unitary process (Merkle & Van Zandt, 2006; Pleskac & Busemeyer, 2010; Ratcliff & Starns, 2009, 2013; Van Zandt & Maldonado-Molina, 2004), our findings suggest that models of confidence should be able to account for carryover effects.

Fourth, we built a novel model for confidence judgments that explains the recovery of the confidence judgment process following an interruption. The model is driven by two parameters. The first is a scaling parameter which accounts for non-decisional processes. The second parameter is the amount of associative activation between elements. This second parameter is theoretically important because it suggests that in procedural tasks, confidence is sensitive to changes in activation.

Fifth, the comparison of the *assoc* parameter for choice RT and confidence strongly suggests that confidence happens after choice. Recall the predictions of the decisional and post-decisional theories of confidence. Decisional theories claim that confidence emerges as a result of the primary choice and that confidence is made available at the same time. Post-decisional models claim that additional information is collected about the primary choice and confidence is formed after a decision is made. Therefore, decisional models would predict that the *assoc* parameter used to model choice RT and confidence would be the same for both because they emerge at the same time. Post-decisional models predict that the activation in the *assoc* parameter would be lower for confidence than for choice RT because the confidence decision occurs after choice RT.

In our study we find that the *assoc* parameter is significantly less for confidence than it is for choice RT. This finding supports the post-decisional theories of confidence that say confidence is rendered after a decision is made. We believe that the *assoc* parameter is lower because the goal used to make the confidence judgment has undergone decay from when the same goal was measured earlier to make the primary choice.

Calculating activation through modeling has some unique benefits for measuring cognitive processes that are otherwise difficult to investigate empirically. For example, in Dotan et al., (2018) the authors offer an alternative explanation of their data. According to some views of confidence processing, their data could be interpreted as very fast and discrete post-processing. By this view, participants make decisions and confidence judgments several times before rendering a final judgment. However, it would be very difficult empirically to disentangle online confidence from rapid post-decision processing. As was the case in our study, modeling could be a helpful tool to help investigate such a hypothesis.

Acknowledgments

This work was supported by the Office of Naval Research to JGT. The views and conclusions contained in this document should not be interpreted as necessarily representing the official policies of the U. S. Navy.

References

Aguiar, N., Zish, K., McCurry, J. M., & Trafton, J. G. (2016). Interruptions Reduce Performance across All Levels of Signal Detection When Estimations of Confidence are Highest. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 60, pp. 254–258). SAGE Publications.

Altmann, E. M., & Trafton, J. G. (2007). Timecourse of recovery from task interruption: Data and a model. *Psychonomic Bulletin & Review*, *14*(6), 1079–1084.

Altmann, E. M., Trafton, J. G., & Hambrick, D. Z. (2014). Momentary interruptions can derail the train of thought. *Journal of Experimental Psychology: General*, *143*(1), 215.

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.

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language*, *38*(4), 341–380.

Edwards M. B., & Gronlund, S. D. (1998). Task interruption and its effects on memory. *Memory*, *6*(6), 665–687.

Baranski, J. V., & Petrusic, W. M. (1998). Probing the locus of confidence judgments: experiments on the time to determine confidence. *Journal of Experimental Psychology: Human Perception and Performance*, *24*(3), 929.

Bradley, J. V. (1968). *Distribution-free statistical tests*.

DeSoto, K. A., & Roediger, H. L. (2014). Positive and negative correlations between confidence and accuracy for the same events in recognition of categorized lists. *Psychological Science*.

Donaldson, W. (1996). The role of decision processes in remembering and knowing. *Memory & Cognition*, *24*(4), 523–533.

Dotan, D., Meyniel, F., & Dehaene, S. (2018). On-line confidence monitoring during decision making. *Cognition*, *171*, 112–121.

Gillie, T., & Broadbent, D. (1989). What makes interruptions disruptive? A study of length, similarity, and complexity. *Psychological Research*, *50*(4), 243–250.

Green, D. M., & Swets, J. A. (1966.). *Signal Detection Theory and Psychophysics*. New York City, New York: Wiley.

Heath, R. A. (1984). Random-walk and accumulator models of psychophysical discrimination: a critical evaluation. *Perception*, *13*(1), 57–65.

Juslin, P., & Olsson, H. (1997). Thurstonian and Brunswikian origins of uncertainty in judgment: a sampling model of confidence in sensory discrimination. *Psychological Review*, *104*(2), 344.

Laming, D. R. J. (1968). Information theory of choice-reaction times.

Lindstrom, M. J., & Bates, D. M. (1990). Nonlinear mixed effects models for repeated measures data. *Biometrics*, 673–687.

Link, S. W., & Heath, R. A. (1975). A sequential theory of psychological discrimination. *Psychometrika*, *40*(1), 77–105.

Merkle, E. C., & Van Zandt, T. (2006). An application of the poisson race model to confidence calibration. *Journal of Experimental Psychology: General*, *135*(3), 391.

Pleskac, T. J., & Busemeyer, J. R. (2010). Two-stage dynamic signal detection: a theory of choice, decision time, and confidence. *Psychological Review*, *117*(3), 864.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, *85*(2), 59.

Ratcliff, R., & Starns, J. J. (2009). Modeling confidence and response time in recognition memory. *Psychological Review*, *116*(1), 59.

Ratcliff, R., & Starns, J. J. (2013). Modeling confidence judgments, response times, and multiple choices in decision making: recognition memory and motion discrimination. *Psychological Review*, *120*(3), 697.

Resulaj, A., Kiani, R., Wolpert, D. M., & Shadlen, M. N. (2009). Changes of mind in decision-making. *Nature*, *461*(7261), 263.

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.

Van Zandt, T., & Maldonado-Molina, M. M. (2004). Response reversals in recognition memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *30*(6), 1147.

Vickers, D. (1970). Evidence for an accumulator model of psychophysical discrimination. *Ergonomics*, *13*(1), 37–58.

Vickers, Douglas. (2001). *Where Does the Balance of Evidence Lie with Respect to Confidence?*

Vickers, Douglas. (2014). *Decision processes in visual perception*. Academic Press.

Wixted, J. T., & Mickes, L. (2010). A continuous dual-process model of remember/know judgments. *Psychological Review*, *117*(4), 1025.

Zish, K., Hassanzadeh, S., McCurry, J. M., & Trafton, J. G. (2015). Interruptions can Change the Perceived Relationship between Accuracy and Confidence. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 59, pp. 230–234). SAGE Publications.

Author Index

Keyword Index