Cognitive Twin: A Personal Assistant Embedded in a Cognitive Architecture

Sterling Somers[1], **Alessandro Oltramari and Christian Lebiere**[1]
[1]Psychology Department, Carnegie Mellon University, Pittsburgh, PA
Bosch Research and Technology Center, Pittsburgh, PA
{sterling@sterlingsomers.com,Alessandro.Oltramari@us.bosch.com, cl@cmu.edu}

Abstract

This paper presents an analysis of a cognitive twin, implemented in a cognitive architecture. The cognitive twin is intended to be a personal assistant that learns to make decisions from your past behavior. In this proof-of-concept case, we have the cognitive twin select attendees to a party, based upon what it has learned (through ratings) about an agent's social network. We evaluate two versions of a model with respect to rate of change in the social network, the noise in the rating data, and the sparsity of the data.

Keywords: cognitive architectures; personal assistants; act-r;

Introduction

Smart recommendation systems, automated alerts, digital personal assistants, and the like have become ubiquitous in our everyday lives. From speakers in our living rooms, watches on our wrists, and, of course, phones in our pockets, we are constantly being notified and updated with information that, at times, can even be helpful. Beyond these current efforts, we envision a cognitive twin: an automated personal assistant that knows the kinds of decisions you would make and uses that knowledge to carry out tasks in the digital world on your behalf. We instantiate our cognitive twin in a cognitive architecture that exhibits human-like cognitive constraints that, for better or worse, can result in biases such as recency and frequency effects. We aim to show that cognitive constraints like these can actually be beneficial in a dynamic environment.

Existing technology requires large data sets which often includes aggregate data from multiple people. There is no doubt that useful inferences can be made by learning from aggregate data, however, there are limits in of the effectiveness of big-data approaches: sometimes the choices you would make are not the same as the average person. Furthermore, these approaches gather data about everyone in a central location which raises concerns about privacy. In our approach, we advocate learning about you from your own data. We cast your behavior as decision that had desirable or undesirable outcomes. That information is stored in the memory of your cognitive twin and is used to make future, similar decisions. Since the decision maker whom we are attempting to model is a cognitive agent, with limited cognitive capacities, we instantiate our model in cognitive architecture that is used in the cognitive sciences to make scientific models of the human mind. The resultant model is human-like and is personalized to make decision for you, negotiate decisions with other cognitive twins, without sharing your data, and without storing your data on a central repository.

In this work we evaluate a prototype cognitive twin developed in the cognitive architecture, ACT-R (Anderson, 2007) using instance-based learning (Gonzalez, Lerch, & Lebiere, 2003). Data is generated from a discrete action simulation of a population of simulated agents who carry out tasks. For each artificial agent in the simulation, a log of their activity is generated. We then use these logs as input to generate a cognitive twin for each simulated agent. In this context, the cognitive twin, therefore, is a simulation of a simulation. As the simulation itself is a prototype, we do not assume a high degree of veridicality. Instead, we use the simulation as a data generator, and evaluate the cognitive twin with respect to that world.

In a machine learning context, it may, at first glance, seem like an misguided choice to use an architecture that is specifically designed to have human-like cognitive constraints. A learning model that, in effect, will 'forget' data over time may seem like a waste of data. However, in a dynamic environment, data can become outdated, as targets change, you might not want to make the same decision you made in the past, even if that decision had a positive outcome. In the present study we test the model's ability to deal with change in two ways. First, directly, we modify the rate of change in the simulation (α) and expect the model to do well with moderate rate of change. Second, we test how many weeks worth of simulated data we use in the model. By reaching back, n-weeks, we hope to show that the learning mechanism gracefully degrades old data; as well as being able to show that the model is robust to few learning examples.

In the spirit of testing real-world-like scenarios, we also test to see how robust the model is to noisy data and sparse data. We test the sensitivity to noise directly by modifying the noise parameter in the simulation (σ). We test sparsity in two ways: first, by using n-previous-weeks worth of data (same as above); and second, by probabilistically controlling how much simulated data is 'seen' by the model (effectively removing data probabilistically).

Background

In previous work (Somers, Oltramari, & Lebiere, 2020) we tested two versions of the cognitive twin, central and dis-

tributed, against one another and against the simulation in their respective success at inviting guests to a 'dinner party'. In this section we describe the dinner-party-planning scenario and the simulation.

Dinner Party Planning

Think about the knowledge that goes into something as seemingly simple as planning a dinner party. You need a place, a time, food, and friends. The time has to suit the attendees (scheduling) and should be during 'dinner' (knowledge), the food should suit the attendees (dietary restrictions), and the people who attend should all get along (unless you are hosting a fight club). Presumably, when you carry out the task of planning a dinner party, you do so partly from knowledge about how to carry-out the task (task-knowledge), partly from things you have learned (who you like, what foods people eat), and partly from communication with other attendees (what food they eat and who they like). Task knowledge can be difficult to learn from data because those processes are often communicated at least in part explicitly as instructions, in a process known as Interactive Task Learning (Laird et al., 2017). We advocate a hybrid approach in which we combine structured knowledge with and data-driven knowledge.

In our previous (and present) work we evaluate our model on its ability to select party guests based on past experiences with them and communicating with other cognitive twins (in distributed model). The simulation assumes a social rating system: when simulated agents interact with one another, they provide a 'rating' of that interaction. That rating is then used to update their social link (their friendship). We describe the simulation, the social interactions, and the rating system more thoroughly below.

Simulation

To generate data, we implemented a discrete simulation. While the simulation, itself, is currently under development, we believe it has enough complexity to generate data to test an early prototype of the cognitive twin. We do not assume that the simulation is veridical on any meaningful dimension but use it, instead, to generate data in lieu of human data. As we progress, we fully intend to develop the simulation further, however, in this analysis we are more concerned about evaluating the cognitive twins, with respect to properties we believe would effect its performance due to underlying cognitive assumptions.

The simulation is comprised of a 'world', populated by agents going about their daily lives. In our previous work our world was comprised of 100 agents, whereas in this analysis, we have doubled the number of agents to 200 (largely to add stability to social networks). Importantly, the agents in the simulation are distinct from their cognitive twins. The agents in the simulation represent 'real' humans (in lieu of real human data) and the twins use the data they generate to plan party attendees. The simulation creates a population of agents by first generating families with sizes controlled by a weighted distribution (parameter). A family is generated, populated, and then new families are generated and populated until the desired population has been reached. Although the analysis we do in this paper is with regards to social interactions, social dynamics between family members are not yet simulated any different than social interactions with other members of the population. We hope to account for differences in social dynamics where appropriate in future work.

Since our proof-of-concept scenario is dinner-party planning, we have developed the simulation to include: social interactions (to develop social links between agents), daily activities (scheduling concerns), and food consumption (to model dietary restrictions). Given space constraints, we will only discuss social interactions (the main focus of this paper).

Social Interactions Each agent in the simulation has both incoming and outgoing social links, that range between -1 (extreme dislike) to +1 (extreme friendliness), to other agents. When a world is generated, the social links are selected from a truncated normal distribution. The mean and standard deviation of the distribution are free parameters but currently set to a mean of 0.4 (under the assumption that social links are generally positive) and a standard deviation of 0.5 (wide enough that negative relationships can still occur). The social links are independent and, as a result, can be asymmetric (i.e. one agent may favor another but may not be favored in return). While negative values represent dislike and positive values represent positive feelings, we explicitly divide positive links into two classes: positive relationships (links greater than zero but less than 0.50) and friends (links greater than or equal to 0.50). In our previous work we set that split at 0.75 but due to changes in the simulation (modified how links change), we are able to better track changes in friendship circles over longer periods of time, by including more people. The initial seeding of social links is used during run-time when setting up social interactions. Roughly, friends become more likely to interact and enemies less likely, as described below. Social interactions (parties) are created during run-time at the start of each simulation day. Each party has a host and each host is selected randomly from the population. The parameter p is set as a proportion of the total population from which the hosts are selected. In our previous work and in the current work we used a value of 0.25 to ensure that we are generating sufficient data to test the model. Modifying this parameter would effect the rate of change in the simulation but we have kept it steady in favor of a parameter that affects the change more directly. Once we have selected hosts for the parties, attendees are selected. Attendees to each party are selected based on their connection to the host. The outgoing links from each center, to each individual in the population is transformed into a weighted probability distribution such that stronger links result in higher likelihood of being selected. The outgoing links from the host are transformed with the following considerations: a) links lower than a minimal value of -0.1 (a free parameter) are excluded; b) a small bias (additive, 0.3, also a free parameter) is added to weights for connections above the 'friend' threshold. These parame-

ters were set qualitatively in previous work to produce steady world statistics.

Even though 25% of the population are chosen as party hosts, not all hosts will result in a social interaction. Once the attendees are selected, the party has to be scheduled and, due to scheduling conflicts, prospective parties may be canceled. For each potential party, a minimum party size is selected randomly from a truncated normal distribution (a free parameter). If the party does not meet that minimum size, the social interaction is canceled. Social interactions are resolved in a queue, which is ordered by the original selection of hosts. Attendees who become committed to a party that is resolved early in the queue may be too busy to attend parties further in the queue. Finally, it is worth noting that although people with large negative connections with the host will not be scheduled by the host, people with negative links could find themselves at someone else's party. The simulation does not try to maximize the overall average connection between guests.

During an interaction, all agents in attendance receive an interaction score with all other guests. In the current work, the interaction score is: the mean links between the agents (incoming and outgoing) plus a randomly selected score value. The score values are selected from a truncated normal distribution between -1 and 1 and the shape of that distribution is a free parameter. We simplify this work from previous work, setting the mean of that distribution to 0.0 (instead of a positive value). In this work, we modify the parameter, σ, which represents the standard deviation of the score value distribution. We consider higher levels of σ as higher levels of noise in the ratings. In the analysis below, we test the robustness of the models to different levels of σ. The final score (mean + score value) is truncated to a range of -1 to 1. This form of scoring is an update from previous work. The social interaction scores are independent: agent A will score the interaction with agent B differently than B will score the interaction with A.

Change in Social Networks The social interactions are the means through which social networks change within the simulation. In this work we have updated how the social links are updated in response to social interactions. We introduce a new parameter, α to represent a rate of change in the simulation that we then systematically modify to test our models against. Social links between agents are updated with the following equation: $L_t = \alpha \cdot L_{t-1} + 1 - \alpha * score_t$, where L_t is the outgoing link at time t and *score* is the interaction score. Higher α values should result in small changes in the networks (because your links are only marginally affected by interaction score), whereas small values of α should result in higher rates of change in social network.

In the present work, we test the robustness of the model at different values of both α and σ. By modifying these two parameters we create low- and high-change simulation conditions (α), and low- and high-noise conditions (σ), creating 4 categories: low-change-low-noise, low-change-high-noise, high-change-low-noise, and high-change-high-noise.

Simulation Analysis

Figure 1 presents different measures of the simulation at each of the four parameters settings: $\alpha = \{0.95, 0.25\} \times \sigma = \{0.25, 0.75\}$.

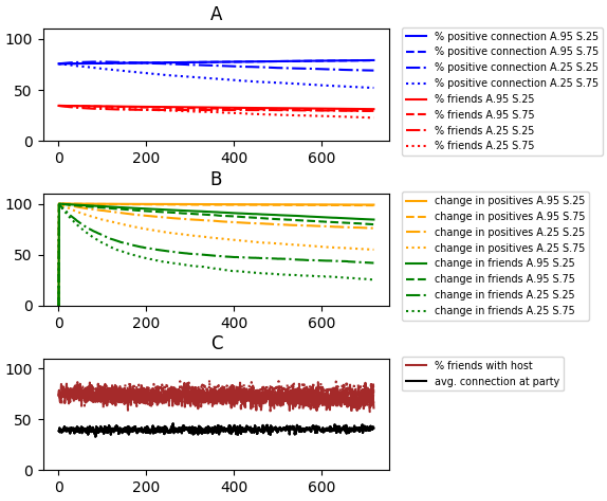

Figure 1: Summary measures of the simulation at different parameters of α and σ. A) shows the world averages of percent positive connections at each parameter settings (blue) and the world average percent of friends (red). B) Illustrates the change in the network from simulation start through two years for both positive connections (orange) and and friendships (green). C) Illustrates the the measures of party success: percent of people who are friends with the host (brown), and the average link between party attendees (black).

Figure 1-A shows the world average percent of positive connections over two years of simulation (blue) and the world average percent of friends, i.e. positive connections greater than 0.50, (red). This plot suggests that, other than $\alpha = 0.25, \sigma = 0.75$n (high-change-high-noise), changes in social networks are not due to a net loss positive connections. The net loss of positive connections (dotted blue line), demarcates a quality boundary, and, therefore provide a good point to limit the adjustment of the α and σ parameters. We expect, in the high-change-high-noise condition, that the cognitive twin would score lower due to overall loss in positive connections.

Figure 1-B illustrates the change in the social networks from time zero through two years of data. Orange lines denote changes in networks of positive connections, while the green lines denote changes in networks of friends (social links greater that 0.50). The change in the networks decay as you might expect: an increase of change with a smaller α and an increased change with increasingly noisy ratings. And, of course, the effect combines, with the greatest rate of change occurring with changes in both α and σ.

Finally, Figure1-C shows the scoring the simulation gets with respect to the percentage of guests who are friends with the host (brown) and the average connection between all guests at a party (black). All parameters are plotted but, due to the fact that those values are somewhat noisy, and the overlapping lines, there is no appreciable difference at different parameter settings. However, they do provide a ball-park estimate how well the simulation does on those measures (the same measure of the model, below). These are the measures we use to evaluate the models.

Cognitive Twin Model

At a high-level, the cognitive twin is an automated decision maker. It uses your data to make the kinds of decisions you would make. In the party-planning scenario, your cognitive twin is intended to do some of the leg-work of party planning for you. The cognitive twin was developed in ACT-UP (Reitter & Lebiere, 2010), a toolkit implementation of ACT-R, developed to make it easier to integrate with simulations, with limited footprint, and network-size scaling requirements. It leverages the key equations for declarative memory, reflecting tradeoffs in recency and frequency in generalizing patters of user activity. Finding the most compatible set of guests relies on ratings of past social activity. The ratings generated by the simulation are represented in the model as chunks in memory consisting of: a) the agent with whom the user interacted, and b) the rating of the interaction (described above). Each of those chunks have an associated base-level activation that reflects the recency (and frequency if the same rating between the same agent has been provided multiple times) of the ratings. Each chunk activation can therefore be interpreted as the relative importance of that rating among competing ones, and factored accordingly by memory retrieval processes. Specifically, the blending retrieval process (Lebiere, 1999) is used to produce a consensus estimate of social rating between two agents by retrieving and aggregating individual ratings weights according to their activation.

These social compatibility ratings are used in a greedy algorithm that starts with the central user organizing the gathering. The rating of all potential guests are evaluated against the current set of invitees, starting with the host, and the guest with the highest rating is selected. In the centered version of the algorithm, the selection process is simply repeated until the guest list is full. In other words, only the social rating between potential guests and the host are taken into consideration. In the distributed version of the algorithm, however, after each guest is selected, the social ratings between that guest and the rest of the potential guests are added in, and the next guest is selected. Again, the process repeats until the guest list is full. Note that it is possible that this process does not yield an optimal outcome because the sequential selection process could lead to a local optimum. For instance, it could select a guest that has the highest social rating with the current guests but very poor social ratings with all remaining potential guests.

Model Analysis

Not only do we test the models with respect to the α and σ parameters in the simulation, we also analyse how robust the models are data sparsity (weeks, probability), and a base-level learning parameter in architecture.

Base-Level The base-level learning parameter controls the rate of power law decay in memory. Memory chunks always remain present in memory, but their activation gradually decays, unless it is boosted again by another experience or rehearsal, reflecting the power law of practice. Since activation controls the retrieval of chunks, activation decay slows and ultimate prevents access to those memories (forgetting) as well as reduces the salience of the chunks in the blending process. While forgetting seems like an unwanted characteristic in an intelligent social agent, it constitutes an adaptive mechanism to the rate of change in the environment, as hypothesized by the rational analysis of cognition (Anderson, 1990; Anderson & Schooler, 1991). As attitudes change and old events becomes less relevant as time passes, favoring more recent information through gradual decay of older memories provides a rational approach to optimizing one's interactions with the world.

We test the effectiveness of the base-level in a number of ways. First, we test how well it responds to modifications of α and σ (which both increase the rate of change, though for different reasons). Second, by modifying the n-previous weeks worth of data, we include increasingly older data in the model. In the high-change conditions, we would expect old data to naturally degrade the twin performance (if it did not have base-level activation) because the social networks have changed, and agents that have interacted long ago may longer be friendly (and visa versa with enemies). Because base-level decay gracefully reduces the impact of older data, we expect the model to be robust, even as we increase the data to the full two-years. Finally, We also test the base-level parameters (BLL) explicitly, specifically In this analysis we start at a generally accepted level of $BLL = 0.5$, and increase it gradually, to explore how well it responds to the increased rate of change.

We also test the robutness of the models with respect to data quantity in two ways described below: n-previous-weeks and probability.

n-Previous Weeks

In our previous work (Somers et al., 2020), we examined how robust the model was to different amounts of data. The simulation at that time, did not have the same social change function (with parameter α). We run a similar analysis here by running the model but including only the last n weeks of data. Figure -(1) and -(2) shows the results of running the model with 1-, 2-, 4-, 8-, 26-, 52-, and 104-past weeks of data.

As expected, the central version of the model (solid) outperforms the distributed model (dotted) in the percent-

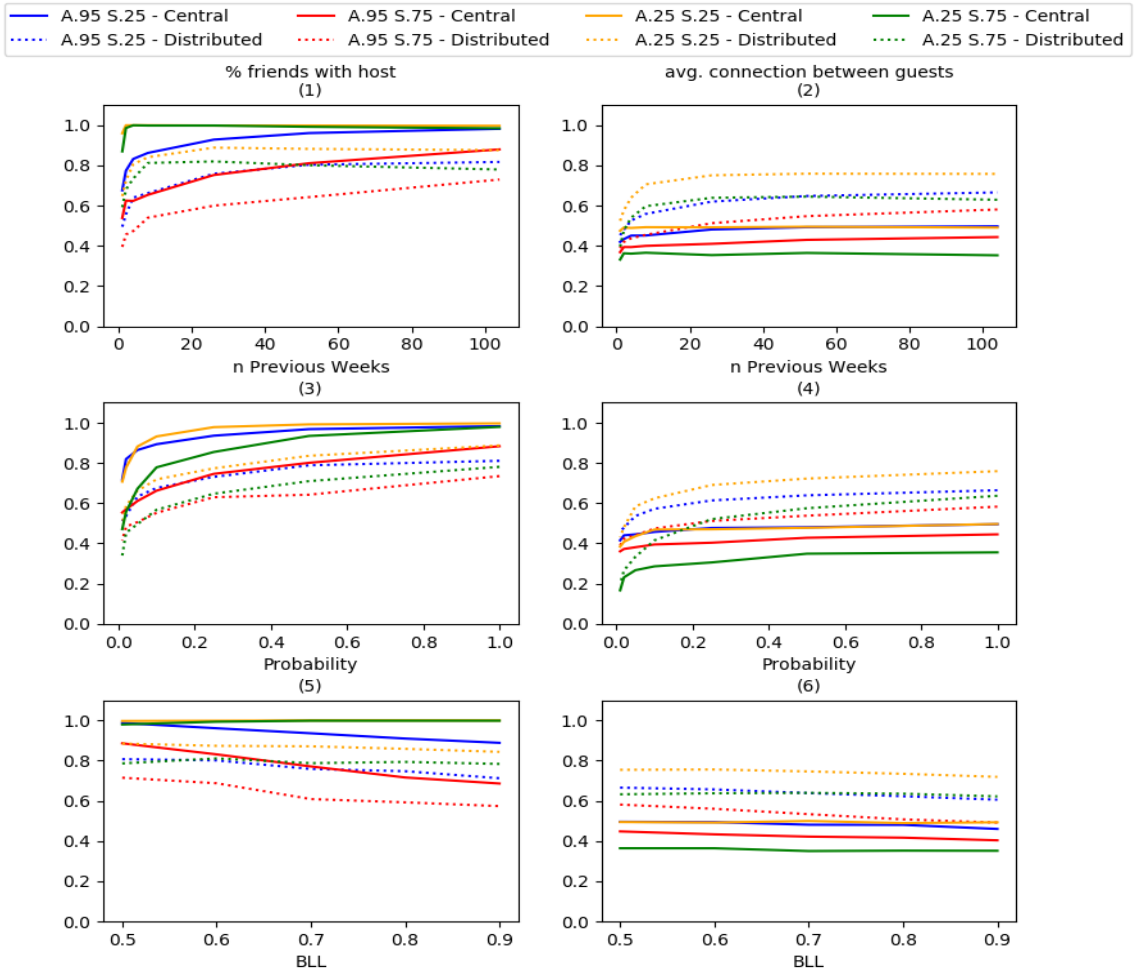

Figure 2: Central (solid) vs. Distributed (dashed) models for each combination of α and σ. The measures are the percent of friends in attendance at a party (left column) and average connection between guests at a party (right column), for *n* previous weeks worth of data (top), probability of including a rating (middle), and the base-level learning parameter of the model (bottom).

friends-with-host (PFH) measure (expected because the central version maximizes for the host, and the distributed maximizes for all attendees). Note, reading left to right on the "n Previous Weeks", although exhibits an increasing amount of data, is actually backwards with respect to time with n-previous being the weeks previous to the end of the simulation. This is important because the simulation is, generally, more stable later in the simulation. In general, the model is quite robust with a small amount of data, with perhaps a slight dip in performance in PFH, in the simulations with the most change (yellow and green), as data reaches back further. In general, the model improves with more data, except in the highest change conditions (yellow and green). In the yellow and green simulation conditions (high change), there is no noticeable improvement in attendees beyond two previous weeks worth of data. Note that although there is no increase, the important trend is limited loss (we want the base-level to naturally degrade old information). In that light, we see a very limited dip in performance at 104 weeks worth of data, suggesting the base-level is mitigating effects of old data in a dynamic environment.

Both the central and distributed model show relatively poor performance in the low-change conditions (blue, red), with few n-previous weeks of data, in the PFH measure. The models exhibit a gradual increase in performance with more data. In these conditions there is very little change to account for, so the base-level appears to over-compensate, benefiting from old data.

In the average-connection-between-guests (AG) measure - (2), the distributed model outperforms the central model (as

expected). The performance of the central model is comparable to the simulation itself (see Figure 1-(c), black lines), with average connection between guests just below the 'friend' threshold (0.50). The central model shows negligible change with increasing number of weeks' data. The distributed models generally improve up to twenty weeks worth of data. There is a fairly strong benefit for the distributed model in the $\alpha = 0.25, \sigma = 0.25$ condition which may indicate a local maximum: a high rate of change (which is taken advantage of by the base-level learning) and a low amount of noise. Beyond 52 weeks, the average connection between friends is above the 'friend' threshold (0.50).

The trend in the AG measure is, overall, similar, showing fairly flat results, suggesting robustness to outdated data in a dynamic environment. There is a slight increase, again, in the low-change conditions (blue, red), for the distributed mode, suggesting, again, that in a static environment, the base-level overcompensates, but recovers with increased data.

Probability

Another measure of data sparsity, one that is probably more naturalistic, is to exclude data probabilistically. We would expect, in reality, to have imperfect participation in our assumed social rating system. In this analysis we encode data in the model probabilistically, simulating agents not providing ratings. Figure -(3) and -(4) illustrates the results of running the model with data probabilities 1.0, 0.5, 0.25, 0.1, 0.05, 0.02, and 0.01. Note that values in this analysis should have the same results at a probability of 1.0 as the weeks analysis had at 104 weeks (they are the same conditions).

Again, central and distributed models preform as expected relative to one another in the respective measures (PFH vs AG). Unlike weeks data, the models typically prefer more data. This difference in amount of data preference is expected because the data is lost evenly with respect to time, so there is no benefit or detriment of the base-level activation. Note that the low-change conditions (blue, red) respond similarly to increased data as in the n-previous weeks. This helps to confirm that the lower performance of the models in those conditions is due to overall data loss, suggesting that those conditions are static, and would benefit from increased data, regardless of how old the data is (as in the n-previous weeks analysis).

The values in both the PFH and AG measures are not too dissimilar from the weeks data, though, notably, the models are not as robust to noise (i.e. red vs blue, green vs orange). The noisy conditions (red, green) perform more poorly in this condition than they had in the n-previous-weeks condition. The lowered performance does make sense, when you consider that in the n-previous-weeks condition, data was removed in blocks, allowing trends to be captured; whereas in the probability condition, data is lost spuriously, amplifying the noise, as might be expected.

Base-Level Learning

In the base-level learning (BLL) analysis, we modify the base-level learning parameter in the architecture from generally accepted range of 0.5, to high value of 0.9 in increments of 0.1. Like the previous analysis, this model plans attendees after the two years of simulation data.

As expected, the central model performs better in the PFH measure (-(5) and -(6)). There is no noticeable change in the PFH scores for the central model, in the highest change conditions (orange, green). The central and distributed model show a decrease performance with an increase in BLL, indicating that increasing the decay rate has an amplified, detrimental effect in a static environment.

In this AG measure, the models are relatively unaffected by the BLL increase. There is not much striking here, other than general inferences, discussed in the next section.

Conclusions

Despite the mildly lower performance in the probability condition, compared to the n-previous-weeks, overall performance of the model is really quite high, especially in comparison to the scheduler in the simulation. In all conditions, other than the high-change-high-noise conditions, the model outperforms the simulation scheduler and in many cases, by a large margin. The distributed model performs especially well scoring reasonably well in both n-previous-weeks and probability conditions, for both PFH and AG measures.

With with respect to change, the model generally prefers a high-change environment (yellow/green vs blue/red) though it responds more poorly with noisy data (e.g. green vs. orange). This is further evidenced in the BLL conditions, with the model scoring more poorly in low-change (blue/red) with increasing BLL parameter. In the low-change conditions, it suggests the cognitive constraints could be a detriment. Looking at Figure 1, however, the low-change conditions are perhaps unrealistically low, showing very little change in social networks over the course of two-years. Future work could explore lowering the BLL parameter in the model to better deal with low-change situations.

Perhaps the most straightforward conclusion from the analysis is that the model is fairly robust to sparse amounts of data. In both n-previous-weeks and data-probability cases, the models perform quite well (in comparison to the simulation). While the models generally show improvement with more data, there is a steep jump (in most cases) with a mild increase in data.

Developing our cognitive twin in a cognitive architecture that is limited by human-like cognitive capacities, though at first glance may seem bizarre, given reliance on big-data approaches, is evidenced to naturally manage data in a dynamic environment. In a dynamic environment, old data eventually may become outdated and detrimental. We have shown that by developing a hybrid learning system, embedded in a cognitive architecture, can be robust to change and data loss.

References

Anderson, J. R. (1990). *The adaptive character of thought*. Psychology Press.

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

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological science*, *2*(6), 396–408.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*(4), 591–635. doi: 10.1016/S0364-0213(03)00031-4

Laird, J. E., Gluck, K., Anderson, J., Forbus, K. D., Jenkins, O. C., Lebiere, C., ... Kirk, J. R. (2017). Interactive task learning. *IEEE Intelligent Systems*, *32*(4), 6-21. doi: 10.1109/MIS.2017.3121552

Lebiere, C. (1999, 06). The dynamics of cognition: An act-r model of cognitive arithmetic. *Kognitionswissenschaft*, *8*, 5-19. doi: 10.1007/s001970050071

Reitter, D., & Lebiere, C. (2010). Accountable modeling in act-up, a scalable, rapid-prototyping act-r implementation. In *Proceedings of the 10th international conference on cognitive modeling, iccm 2010* (pp. 199–204).

Somers, S., Oltramari, A., & Lebiere, C. (2020). Cognitive twin: A cognitive approach to personalized assistants. In *Aaai spring symposium: Combining machine learning with knowledge engineering (1).*