Proceedings of ICCM 2025

23rd International Conference on Cognitive Modelling[1]

Edited by

Alexander R. Hough and Catherine Sibert

---

[1]Co-located with the 58th Annual Meeting of the Society for Mathematical Psychology

Preface

The International Conference on Cognitive Modelling (ICCM) is the premier conference for research on computational models and computation-based theories of human cognition. ICCM is a forum for presenting and discussing the complete spectrum of cognitive modelling approaches, including connectionism, symbolic modeling, dynamical systems, Bayesian modeling, and cognitive architectures. Research topics can range from low-level perception to high-level reasoning. In 2025, ICCM was jointly held with MathPsych—the annual meeting of the Society for Mathematical Psychology. The conference was held at The Ohio State University July 25th to July 29th. An additional, virtual conference was held online from June 16th to June 20th. Submissions from both the in-person and virtual conferences are included in these proceedings. It is available at http://acs.ist.psu.edu/papers/ICCM2025Proceedings.pdf and other mirrors.

Acknowledgements

We would like to thank the Society for Mathematical Psychology (SMP) for their ongoing commitment to the collaboration between our societies. In particular, the MathPsych Conference Chairs (Leslie Blaha, Peter Kvam, and Brandon Turner) and the many officers of the SMP for giving much needed logistical support.

Papers in this volume may be cited as:
Lastname, A., Lastname, B., & Lastname, C. (2025). Title of the paper. In A. R. Hough & C. Sibert (Eds.), *Proceedings of the 23rd International Conference on Cognitive Modelling* (pp x-x). University Park, PA: Applied Cognitive Science Lab, Penn State.

ISBN-13: 978-0-9985082-9-0, published by the Applied Cognitive Science Lab, Penn State.

ICCM Conference Committees

Conference Chairs

Alexander R. Hough	Air Force Research Laboratory
Catherine Sibert	University of Groningen

Program Committee

Amirreza Bagherzadehkhorasani	The Pennsylvania State University
Luis Alberto Barradas Chacón	University of Hildesheim
Hillmer Chona	The Pennsylvania State University
Michael Collins	Wright State University
Taylor Curley	Air Force Research Laboratory
Christopher Dancy	The Pennsylvania State University
Swapnika Dulam	The Pennsylvania State University
Nathan Gillespie	University at Albany, SUNY
Joseph Killian Jr.	Drexel University
Lingyun He	The Pennsylvania State University
Nils Wendel Heinrich	Universität zu Lübeck
Mark Ji	University of Groningen
Mary Alexandria Kelly	Carleton University
Joseph Glavan	Rensselaer Polytechnic Institute
Othalia Larue	Université du Québec à Montréal
Justin Li	Occidental College
Peter Lindes	University of Michigan
Sky Liu	Cardiff University
Junya Morita	Shizuoka University
Kazuma Nagashima	Shizuoka University
David Peebles	University of Huddersfield
Jongchan Pyeon	The Pennsylvania State University
Stefan Radev	Rensselaer Polytechnic Institute
Frank Ritter	The Pennsylvania State University
Nele Russwinkel	University of Luebeck
Dario Salvucci	Drexel University
Florian Seitz	University of Basel
Thomas Sievers	University of Lübeck
Terry Stewart	National Research Council Canada
Anastasia Stoops	University of Illinois at Urbana-Champaign
Eilene Tomkins-Flanagan	Carleton University
Maarten van der Velde	MemoryLab

Table of Contents

Using Evolution to Discover Concept Learning Strategies in a Cognitive Architecture

Dmitry Bennett (D.Bennett5@lse.ac.uk)
Centre for Philosophy of Natural and Social Science, Houghton Street,
London School of Economics, WC2A 2AE, United Kingdom

Noman Javed (N.Javed3@lse.ac.uk)
Centre for Philosophy of Natural and Social Science, Houghton Street,
London School of Economics, WC2A 2AE, United Kingdom

Laura Bartlett (L.Bartlett@lse.ac.uk)
Centre for Philosophy of Natural and Social Science, Houghton Street,
London School of Economics, WC2A 2AE, United Kingdom

Peter Lane (p.c.lane@herts.ac.uk)
Department of Computer Science, College Lane,
University of Hertfordshire, AL10 9AB, United Kingdom

Fernand Gobet (F.Gobet@lse.ac.uk)
Centre for Philosophy of Natural and Social Science, Houghton Street,
London School of Economics, WC2A 2AE, United Kingdom

Abstract

Research into concept formation is concerned with the fundamental question of how we categorise objects into a particular class. This study develops cognitive agents based on the CHREST model that is placed into a genetic programming environment (GEMS), thus generating cognitive strategies. We evolve populations of agents learning by chunking, with each agent incorporating LTM and STM stores, as well as attention mechanisms. The resulting models simulate the seminal concept learning tasks that investigate the learning rate of linearly and nonlinearly separable categories of various complexity (Nosofsky et al., 1994; Shepard, Hovland, & Jenkins, 1961). CHREST's results are compared to both human data and EPAM – a different chunking-based learning model that utilises hand-crafted task-specific strategies. The automatically evolved CHREST strategies produced good fit to the human data in both studies, improving on EPAM's scores by as much as factor of two on some of the categories. These findings fuse human-like concept learning strategies with CHREST's powerful general learning mechanisms, and exemplify GEMS automated discovery of cognitive function.

Keywords: learning; chunking; genetic programming; GEMS; CHREST; learning; long-term memory; short-term memory.

Introduction

Research into concept formation is concerned with the fundamental question of how we categorise objects into a particular class. A key result is that natural categories lack consistent defining features and are better characterized as the outcome of inductive learning of statistical distributions and functions (Braunlich & Love, 2022; Medin & Schaffer, 1978; Nosofsky, 2011; Rosch, Simpson, & Miller, 1976). Psychologists have developed mathematical models that capture the statistical abstraction approach into formalisms of prototypes (which treat concepts as sets of weighted probabilistic features) (Rosch, 1973), exemplars (for deriving concepts from typicality gradients across all specific instances) (Nosofsky et al., 1994), and models based on adaptive learning types of clustering (Braunlich & Love, 2022).

Being widely influential, the exemplar, prototype and clustering theories were incorporated into broader cognitive architectures such as ACT-R and Soar (e.g., Chong & Wray, 2003; Rutledge-Taylor et al., 2012) or, alternatively, augmented with algorithms based on deep learning (Battleday, Peterson, & Griffiths, 2020). This provided rigorous links to how the initial mathematical models may be connected to other and/or broader phenomena – such as language acquisition, problem solving and memory decay. Another approach was to forgo the mathematical models and to build the concept learning theory from the foundations of a cognitive architecture itself (e.g., Richman & Simon, 2002).

In all three cases, it was typical for the models to feature hand-crafted attentional and learning strategies. Indeed, model development requires assumptions regarding the cognitive/psychological mechanisms that are involved in various experimental settings. On the other hand, this often leads to experimenters' bias, with researchers inadvertently overlooking potentially important mechanisms at play.

In this context, the aim of the current paper is threefold. First, we take Newell's (1990) advice and aim to integrate concept learning into a broader theoretical structure. To that

end, we use CHREST – a cognitive architecture – as a foundation for the concept learning models.

Our second aim is to use the GEMS environment (Lane et al., 2022), which is based on Genetic Programming (GP) (Koza, 1992), to create and evolve candidate models. One downside of working with complex cognitive architectures is the laborious process of hand-crafting task-specific strategies that incorporate many interacting parts and processes. The use of GEMS, on the other hand, helps to automatise and speed up this process. The other benefit of GEMS is how it reduces the experimenter's bias when designing and tuning the concept learning models, with the experimenter only deciding on the choice of cognitive architecture (e.g., CHREST, ACT-R, Soar, a mixture of all three, etc.), the initial set of operators and the fitness function that provides the desired goal for the evolving models.

Our last objective is to further bridge the gap between models that have a good fit to human experiments on simple pre-processed categories and models that are used for outright performance on raw complex multidimensional natural concepts. It is common for a model to perform well in one of these domains, but badly in the other. CHREST has recently been shown to perform well in categorising raw natural categories (Bennett & Gobet, 2024), and, given its deep roots in psychology, it is fitting to evaluate how well it models human concept learning data with simple categories.

The Cognitive-Based CHREST Model

CHREST (Chunking Hierarchy and REtrieval STructures) (Gobet, 1993, 2000; Gobet & Lane, 2012; Gobet & Simon, 2000) is a cognitive architecture based on the chunking theory – one of the most deep-rooted theories in cognitive psychology (Chase & Simon, 1973; Gobet et al., 2001; Simon, 1974).

The motivation behind CHREST originated from criticism of the traditionally disjointed approach to cognitive science and its task-specific models (Newell, 1973). Cognitive architectures were presented as a solution to that problem, aiming to integrate theories of various cognitive mechanisms and representations into unitary computational embodiments (Byrne, 2012; Lane & Gobet, 2012b; Newell, 1990).

Consequently, CHREST and related chunk-based models were used to explain and predict behaviour in the acquisition of language (Freudenthal et al., 2016; Jessop, Pine, & Gobet, 2025), verbal learning (Bennett et al., 2024), problem solving (Lane, Cheng, & Gobet, 2000), emotion processing in problem gambling (Schiller & Gobet, 2014), categorisation of novel literature and music pieces (Bennett & Gobet, 2024; Bennett, Gobet, & Lane, 2020), developmental trends and cognitive decline due to ageing (Mathy et al., 2016; Smith, Gobet, & Lane, 2007), expert behaviour (Gobet & Simon, 2000; Richman et al., 1996; Richman, Staszewski, & Simon, 1995; Simon & Gilmartin, 1973), and the list goes on.

The core concept of the chunking theory – a *chunk* – can be defined as a meaningful unit of information constructed from elements that have strong associations between each other. CHREST formalises chunks as nodes in a graph data structure that represents LTM. The process of chunking is used to update the LTM with new data – by either updating the existing nodes (familiarisation) or creating new ones (discrimination) (Gobet, 1993, 2000; Gobet & Lane, 2012; Gobet & Simon, 2000). CHREST's STM structure allows for retrieval of items from the LTM and additional processing – such as linking stimuli across verbal and visual modalities.

The other important feature of the chunking theory is that it uses established experimental data to set the time costs for its core cognitive operations (Simon, 1974; Simon & Gilmartin, 1973). These are also integrated in CHREST. For example, recognition of a pattern requires traversing several nodes (10 milliseconds each), familiarisation (adding information to an existing chunk) takes two seconds, and discrimination (creating a new chunk) takes ten seconds.

For more details on the chunking theory and cognitive architectures see Gobet and Lane (2012).

The Genetic Side – GEMS

GEMS (Genetically Evolving Models in Science) is a meta-modelling program framework designed to semi-automatically develop cognitive models (Bartlett et al., 2023; Lane et al., 2022). These cognitive models consist of operators derived from research literature in psychology. Each operator requires a certain amount of time for execution, and GEMS tracks these timings using internal clocks. As a result, the overall time consumed by a model developed with GEMS can be calculated.

To assess the performance of a model, GEMS simulates the experimental conditions of the original experiment. The accuracy and timings of the simulations are recorded by executing the models over a cognitive architecture, which is CHREST in our case (See Figure 1).

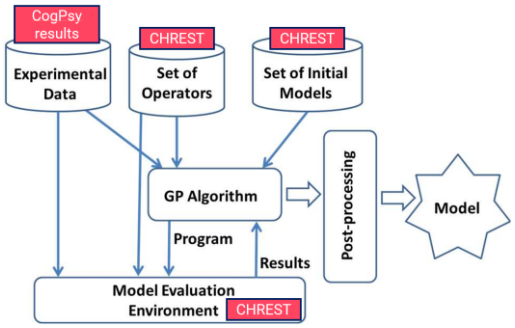


Figure 1. Overview of GEMS and its interaction with CHREST.

Genetic programming employs these fitness values to search the state space of the cognitive models. It acts as a generator and a breeder, creating and combining models following principles inspired by natural selection. This process evolves a population of models, pushing the top-ranked ones into future generations and using them to produce offspring. When an evolutionary run ends, it yields several distinct models with the best fitness scores, which serve as candidate solutions.

The final step involves refining these candidate solutions by removing any non-functional code that became part of

their genetic makeup during evolution. These simplified models simulate the behaviour of human participants in experimental settings and can be considered as the strategies adopted by humans during the experiment.

At the end of a run or multiple runs of GEMS, the system yields a population of good quality solutions. We have incorporated several post-simplification and analysis functionalities into the system to further simplify these models and cluster them based on their similarities (Lane et al., 2022). However, all this is contingent on the requirements of the experiment one is trying to simulate.

Alternative GP approaches also generate cognitive models to simulate behaviours and agentic strategies. However, their underlying cognitive mechanisms are often simple and lack simulations of the LTM and perceptual apparatus (e.g., Bartlett et al., 2023; Gunaratne & Patton, 2022; Pires & Cota, 2015). By contrast, our approach combines GP (in the form of GEMS) with the highly developed cognitive architecture CHREST. This allows us to combine complex psychological mechanisms and structures with the semi-automatic search of the cognitive strategy space, yielding models that reflect the complexity presumably involved in concept learning.

Simulations of concept-learning experiments

As mentioned above, CHREST can categorise raw natural highly multi-dimensional stimuli without bootstrapping to pre-learned knowledge structures (Bennett et al., 2020) – typical examples of the latter include semantic dictionaries (e.g., Lieto et al., 2017), hand crafted chess heuristics (e.g., Lane & Gobet, 2012a), and geology expert devised similarity matrices (e.g., Nosofsky, Meagher, & Kumar, 2022). CHREST's performance was also shown to be similar to a deep-learning model in psychologically important ways – despite the vast differences in mechanisms (Bennett & Gobet, 2024). However, one shortcoming of these studies was their lack of comparison to human data. The current paper aims to address this limitation by testing the psychological plausibility of CHREST's learning mechanisms against the classic human concept learning experiment carried out by Shepard et al. (1961). Despite its apparent simplicity, this experiment has been used as a benchmark in categorisation research for decades and continues to feature in high-impact modern psychology research (Braunlich & Love, 2022; Goodman et al., 2008; Nosofsky, 2011).

The experiment involves establishing learning patterns for six different types of classification, where the same set of eight stimuli have to be split into two classes A and B (see Table 1 for the complete set of category structures). A strong correlation was found between the category types and the learning difficulty for human participants (see Figure 2): Type I is the easiest to learn, followed by Type II, which is generally easier than Types III, IV, and V, which all present similar levels of difficulty. Type VI, on the other hand, proved to be the most challenging, requiring learners to memorize all possible configurations of the stimuli.

The differing learning rates across these six category types have become a litmus test in the field, with computational models expected to account for these results to be considered credible explanations of human category learning. The fast learning of non-linear XOR-like Type II and slow learning of linearly separable Type IV tends to foil models that do not account for human selective attention and localist encoding. For example, deep learning/backpropagation-based models tend to learn Type IV faster than Type II (Kurtz, 2007).

A number of mathematical models provide excellent fit to the human data – for example, SEA (Braunlich & Love, 2022), RULEX (Nosofsky et al., 1994), DIVA (Kurtz, 2007) and rational-rule model (Goodman et al., 2008), with respective r^2 scores being above .95.

There have been a number of instances where the mathematical models above were implemented in broader theories of cognition/cognitive architectures (e.g., Chong & Wray, 2003; Rutledge-Taylor et al., 2012). Another approach has been to hand-craft bespoke concept learning models from within the cognitive architecture. For example, Richman and Simon (2002) applied the EPAM architecture, which was originally developed for modelling verbal-learning and expertise data, to concept formation. Their model had 12 Learn and Respond routines, some of which branched into further subroutines. For instance, "if the current stimulus and the previous stimulus are not members of the same category the hypothesis will be created by comparing the exception to itself instead of by comparing the exception (or non-exception) to the previous stimulus" (Richman & Simon, 2002, p. 11). EPAM model also had an overall good fit to the human data (r^2 = .94 for its average proportions of errors), but, despite the hand-crafted strategies, the human and EPAM error rates differed by a factor of 2 or 3 for categories of Types III – VI (see Figure 4).

Table 1. The six types of categories in Shepard et al. (1961).

Stimulus	Category Types					
	I	II	III	IV	V	VI
[0, 0, 0]	A	A	B	B	B	B
[0, 0, 1]	A	A	B	B	B	A
[0, 1, 0]	A	B	B	B	B	A
[0, 1, 1]	A	B	A	A	A	B
[1, 0, 0]	B	B	A	B	A	A
[1, 0, 1]	B	B	B	A	A	B
[1, 1, 0]	B	A	A	A	A	B
[1, 1, 1]	B	A	A	A	B	A

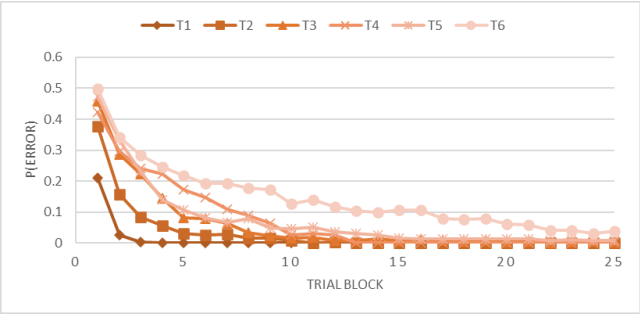


Figure 2. Average probabilities of human errors for each category type in each block of 16 trials. Adapted from Nosofsky et al. (1994).

Table 2. Overview of GEMS operators. Each operator type had a time cost (in milliseconds, ms) as follows: input (100 ms), output (140 ms), LTM (2,000 ms for familiarisation, 10,000 ms for discrimination), and syntax (0 ms).

Operator	Function	Type
PROG-X	a sequence of 2, 3 or 4 subprograms	Syntax
REPEAT2	repeats a subprogram 2 times	Syntax
ATTEND-STIMULUS	place the stimulus value into input slot 1	Input
ATTEND-RESPONSE	place the response value into input slot 2	Input
REC-AND-LEARN-ST	calls CHREST's recognise-and-learn-pattern function to learn a stimulus	LTM
REC-AND-LEARN-RES	calls CHREST's recognise-and-learn-pattern function to learn a response	LTM
RECOGNISE-ST	calls CHREST's recognize-pattern function to locate a pattern in long-term memory	LTM
LEARN-AND-LINK	calls CHREST's learn-and-link-two-patterns function to associate stimulus with response	LTM
RESPOND	retrieve the linked pattern using the stimulus and assign it to the model's output slot	Output
WAIT-X	advances model-clock (in ms): 1000 or 2000	Time

The Present Study

The current study aims to simulate the results of the human concept learning experiment reported by Shepard et al. (1961) and replicated by Nosofsky et al. (1994). The contribution of the current study is in using CHREST as a foundation for concept learning models and fusing it with the evolutionary search offered by GEMS. Integrating concept learning into the CHREST cognitive architecture will potentially connect concept learning with other behaviours that have been simulated with chunking mechanisms – such as problem solving, language acquisition and developmental decay due to ageing (to mention but three). The incorporation of GEMS, on the other hand, will allow us to remain agnostic with regard to the category-specific attentional and learning strategies, while still optimizing the fit of the models with human data. This will minimise our theoretical bias towards, for example, exemplar, prototype or rule-based approaches to concept learning. Lastly, it will also let us check if the previously reported powerful general learning capability of CHREST – which, similar to deep learning, allows it to learn from raw complex natural data without bootstrapping – is consistent with our current understanding of human concept learning.

Method

Training and Testing

The human data were taken from Nosofsky et al. (1994), who replicated the training and testing procedure used by Shepard et al. (1961). Each trial consisted of (a) a stimulus being presented to the model, (b) the model classifying the stimulus into category A or B, and (c) feedback on the accuracy of the response. The model's performance was monitored over 25 blocks of learning trials (there were 8 trials in the 1st and 2nd blocks, and 16 trials in subsequent blocks). While the original experiment was self-paced, we imposed a 30 second limit as the maximum time that a model was allowed to spend per trial (it could also choose shorter presentation time durations).

In order to generate concept learning strategies automatically, CHREST was integrated with GEMS. Thus, instead of a single model, a population of 500 models were simultaneously evaluated on the task. Each model initially consisted of random program trees made from cognitive operators. The operators included `attend stimulus`, `attend feedback`, `recognise stimulus`, `recognise and learn stimulus`, `recognise and learn response`, `learn the link between the stimulus and the response`, `repeat operation`, and `wait for up to 30 seconds` (see Table 2). The best performing models were crossbred (with a mutation rate of .20) for 100 generations. Since each category type requires a different cognitive strategy (Rehder & Hoffman, 2005), the models were trained on "per category type" basis. The fitness was the sum of categorisation errors for an individual CHREST model. At the end of the run, GEMS automatically picked around 300 models that, when averaged, had the best fit to the human data (also averaged).

Please see https://github.com/Voskod/GEMS for the code and the best models for all the experimental conditions.

Results

The results of the evolved CHREST concept learning models are presented in Figures 3 and 4. CHREST achieved good fit to the human data in Nosofsky's concept learning study. Looking at the six category types individually (from Type I to Type VI), the corresponding r^2 values for P(Error) across learning trial blocks are .89, .89, .94, .80, .97, and .90 (see Figure 3). The overall r^2 for the average proportion of errors across all category types is .97 (see Figure 4).

The average proportion of errors per category type for CHREST follows the same ordering as humans in the study by Nosofsky et al. (1994). CHREST's error rates are closer to those of humans when compared to the results of the previous chunking-based hand-crafted EPAM model.

CHREST does worse than people on the very first block of learning (with human average error rate across all category types being .35, while that of CHREST's being .64) – we suspect that this is due to the differences in self-paced versus time-limited stimuli presentation in human and simulated

experiments, respectively, as opposed to a fundamental deficiency of the models' learning. Indeed, some individual CHREST models displayed human-like errors rates on their first trial (e.g., around .25 for Type I).

A sample strategy for the Type I category is presented in Figure 5; it will be briefly explained in the Discussion section.

We should note that we are reporting only the results with the best group of models, with there being many groups of models that achieved similar fit.

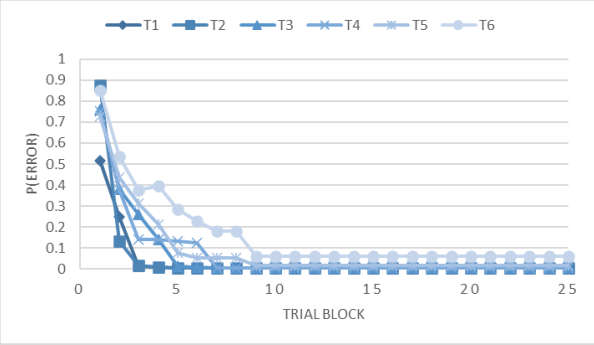


Figure 3. Average probabilities of CHREST models' errors for each category type in each block of 16 trials.

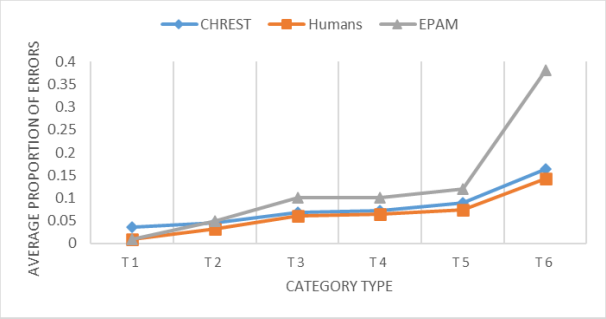


Figure 4. The comparison of average proportion of errors per category type for humans, CHREST and EPAM. Human data are from Nosofsky et al. (1994), and EPAM data are from Richman and Simon (2002).

```
(PROG4
   (PROG4
      (RESPOND)
      (WAIT-2000)
      (PROG2
         (WAIT-1000)
         (PROG4
            (ATTEND-STIMULUS)
            (RECOGNISE-ST)
            (WAIT 1000)
            (WAIT-1000)))
      (RESPOND))
   (REPEAT2
      (RECOGNISE-ST)
      (REC-AND-LEARN-RES))
                              ... ⮭
...
(PROG4
      (WAIT-2000)
      (ATTEND-STIMULUS)
      (REC-AND-LEARN-RES)
      (RESPOND))
(REPEAT2
      (LEARN-AND-LINK)
      (REPEAT2
         (REPEAT2
               (LEARN-AND-LINK)
               (WAIT-2000))
         (PROG2
            (REC-AND-LEARN-ST)
            (ATTEND-RESPONSE)))))
```

Figure 5. One of the strategies in the final population of models that learnt to classify Type I category stimuli.

Discussion

There are three important contributions of the current study. First, CHREST, a modern chunk-based cognitive architecture, has passed the litmus test of simulating the seminal concept learning experiment carried out by Shepard et al. (1961). CHREST largely replicated the pattern of results found in human learning: e.g., learning non-linear Type II faster than the linearly separable Type IV, and generally producing proportional error rates similar to those of humans. This differentiates CHREST from pure machine-learning algorithms based on deep learning, whose pattern of learning differ significantly from humans'; e.g., taking longer to learn nonlinear XOR/Type II than the linearly separable Type IV category (Kurtz, 2007).

The second contribution is in moving away from hand-crafted learning strategies that were used in previous research. Indeed, the authors of an earlier chunk-based cognitive architecture lamented their already complex hand-crafted procedures as being not complex enough to address the relatively poor Type VI performance and suggested adding parameters for switching from rote learning to hypothesis formation (Richman & Simon, 2002). In contrast, GEMS made it possible to automatically develop multiple cognitive strategies that solved the Type VI problem in human-like fashion.

The interpretability of the models (programs) that were evolved by GEMS is worth stressing. These are not "black boxes" and stem directly from the definitions of the cognitive operators of the underlying architecture (this is unlike the Bayesian and artificial neural network approaches). For example, let us consider the simple cognitive strategy in Figure 5 for the Type I condition. Skimming through the strategy, we understand that it is applied for each presentation of a stimulus. The model initially focuses on the stimulus, learns the stimulus, and after responding with its category guess, it then reevaluates the stimulus and feedback/response. Deeper level analysis into timings (which are important as LTM's discrimination and familiarisation processes cost time) and the order of cognitive operators is also possible.

The aspects of human-like learning and interpretability are even more intriguing as CHREST *is* otherwise comparable to

deep learning in its ability to learn from raw multidimensional naturalistic data (Bennett & Gobet, 2024) – an area where traditional psychological models of concept learning have struggled (when not bootstrapped by pre-processed data or augmented with deep learning to do the heavy lifting of feature learning). In this sense, CHREST reversed the traditional trajectory of simulating artificial category experiments with the aim of discovering mechanisms that generalise to naturalistic data. This forms the third key contribution of the current study.

One possible critique of our modelling approach is "overfitting" – when overly complex models attain near-perfect results on training sets but struggle to generalize effectively to data outside of the simulated scenarios. In our view, the CHREST models did not show these signs, as the resulting programs were relatively concise, psychologically meaningful and traced from the start of training. This may be further investigated with CHREST's simulations of other concept learning experiments.

Another potential issue is more epistemological. Indeed, it may be argued that an attempt to eliminate the researcher's bias through search is an illegitimate move that violates the central tenet of falsificationism (Popper, 1959). Moreover, theories cannot emerge from data, as data are always particular and never general. Our answer to these criticisms is to stress the *semi*-automatic nature of our models. GEMS's search through the space of candidate strategies is indeed fully automatic/atheoretic. However, the underlying CHREST and GEMS mechanisms, as well as the operators chosen, do form our theoretical commitments. Thus, our overall approach reduces – rather than eliminates – theoretical bias towards, e.g., exemplars, prototypes or specific chains of counterfactual reasoning. The combination of theory and search allowed us to show that, for example, despite previous studies being unable to find chunking-based strategies that were complex enough to offer a good fit to Type VI categorisation performance, such strategies do exist and that the underlying chunking theory is fundamentally compatible with our understanding of human concept learning.

To conclude, our study integrates concept learning into a powerful general learning chunk-based cognitive architecture and presents a method to automate the discovery of task-specific cognitive processes.

References

Bartlett, L., Pirrone, A., Javed, N., Lane, P., & Gobet, F. (2023). Genetic programming for developing simple cognitive models. In F. K. A. M. Goldwater, B. K. Hayes, & D. C. Ong (Ed.), *Proceedings of the 45th annual conference of the cognitive science society*. Sydney, Australia.

Battleday, R. M., Peterson, J. C., & Griffiths, T. L. (2020). Capturing human categorization of natural images by combining deep networks and cognitive models. *Nature Communications, 11*.

Bennett, D., & Gobet, F. (2024). Cognitive chunks, neural engrams and natural concepts: Bridging the gap between connectionism and symbolism. In *2024 ieee 6th international conference on cognitive machine intelligence (cogmi)* (pp. 217-225). Washington, DC.

Bennett, D., Gobet, F., & Lane, P. (2020). Forming concepts of Mozart and Homer using short-term and long-term memory: A computational model based on chunking. In S. Denison, M. Mack, Y. Xu, & B. C. Armstrong (Eds.), *Proceedings of the 42nd annual conference of the cognitive science society* (pp. 178-184). Toronto.

Bennett, D., Javed, N., Lane, P., Bartlett, L., & Gobet, F. (2024). Genetically evolving verbal learner: A computational model based on chunking and evolution. In *22nd international conference on cognitive modeling (iccm) society for mahtematical psychology*. Tilburg, NL.

Braunlich, K., & Love, B. (2022). Bidirectional influences of information sampling and concept learning. *Psychological Review, 129*(2), 213-234. doi:10.1037/rev0000287

Byrne, M. D. (2012). Unified theories of cognition. *WIREs Cognitive Science, 3*(4), 431-438.

Chase, W. G., & Simon, H. (1973). Perception in chess. *Cognitive Psychology, 4*, 55-81.

Chong, R., & Wray, R. (2003). Rulex-em: Incorporating exemplars and memory effects in a hypothesis-testing model of category learning. In *European cognitive sicence* (pp. 79-84). Osnabruck, Germany.

Freudenthal, D., Pine, J., Jones, G., & Gobet, F. (2016). Developmentally plausible learning of word categories from distributional statistics. In D. Papafragou, D. Grodner, D. Mirman, & J. Trueswell (Eds.), *38th annual conference of the cognitive science society*. Austin, TX.

Gobet, F. (1993). A computer model of chess memory. In W. Kintsch (Ed.), *Fifteenth annual meeting of the cognitive science society* (pp. 463-468): Erlbaum.

Gobet, F. (2000). *Discrimination nets, production systems and semantic networks: Elements of a unified framework*. Evanston: The Association for the Advancement of Computing in Education.

Gobet, F., & Lane, P. (2012). Chunking mechanisms and learning. In M. M. Seel (Ed.), *Encyclopedia of the sciences of learning* (pp. 541-544). New York: NY: Springer.

Gobet, F., Lane, P. C., Croker, S., Cheng, P. C., Jones, G., Oliver, I., & Pine, J. M. (2001). Chunking mechanisms in human learning. *Trends in Cognitive Sciences, 5*(6), 236.

Gobet, F., & Simon, H. (2000). Five seconds or sixty? Presentation time in expert memory. *Cognitive Science, 24*, 651-682.

Goodman, N. D., Tenenbaum, J. B., Feldman, J., & Griffiths, T. L. (2008). A rational analysis of rule-based concept learning. *Cognitive Science, 32(1)*, 108–154.

Gunaratne, C., & Patton, R. (2022). Genetic programming for understanding cognitive biases that generate polarization in social networks. In *Proceedings of the genetic and evolutionary computation conference companion* (pp.

546–549). Boston, MA: Association for Computing Machinery.

Jessop, A., Pine, J., & Gobet, F. (2025). Chunk-based incremental processing and learning: An integrated theory of word discovery, vocabulary growth, and speed of lexical processing. . *Psychological Review*.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection.* Cambridge, MA: MIT Press.

Kurtz, K. (2007). The divergent autoencoder (diva) model of category learning. *Psychonomic Bulletin & Review, 14*(4), 560-576. doi:10.3758/BF03196806

Lane, P., Bartlett, L., Javed, N., Pirrone, A., & Gobet, F. (2022). Evolving understandable cognitive models. In *Proceedings of the 20th international conference on cognitive modelling*. Toronto.

Lane, P., Cheng, P. C.-H., & Gobet, F. (2000). CHREST + : Investigating how humans learn to solve problems using diagrams. *AISB Quarterly, 103*, 24–30.

Lane, P., & Gobet, F. (2012a). CHREST models of implicit learning and board game interpretation. In J. Bach, B. Goertzel, & M. Iklé (Eds.), *Artificial general intelligence: 5th international conference* (pp. 148-157). Berlin: Springer.

Lane, P., & Gobet, F. (2012b). A theory-driven testing methodology for developing scientific software. *Journal of Experimental & Theoretical Artificial Intelligence., 24*(4), 421-456.

Lieto, A., Radicioni, D., Rho, V., & Mensa, E. (2017). Towards a unifying framework for conceptual represention and reasoning in cognitive systems. *Intelligenza Artificiale, 11*(2), 139-153.

Mathy, F., Fartoukh, M., Gauvrit, N., & Guida, A. (2016). Developmental abilities to form chunks in immediate memory and its non-relationship to span development. *Frontiers in Psychology, 7*, 201.

Medin, D. L., & Schaffer, M. (1978). Context theory of classification learning. *Psychological Review, 85*(3), 207-238. doi:10.1037/0033-295X.85.3.207

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In W. G. Chase (Ed.), *Visual information processing* (pp. 283-308). New York: Academic Press.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA, US: Harvard University Press.

Nosofsky, R. M. (2011). The generalized context model: An exemplar model of classification. In *Formal approaches in categorization.* (pp. 18-39). New York: Cambridge University Press.

Nosofsky, R. M., Gluck, M. A., Palmeri, T. J., Mckinley, S. C., & Glauthier, P. (1994). Comparing models of rule based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory and Cognition, 22*, 352-369.

Nosofsky, R. M., Meagher, B. J., & Kumar, P. (2022). Contrasting exemplar and prototype models in a natural-science category domain. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 48*(12), 1970-1994. doi:10.1037/xlm0001069

Pires, J. M., & Cota, M. P. (2015). A new learning cognitive architecture using a statistical function and genetic algorithms: An intelligent new e-learning model. In *2015 fifth international conference on e-learning* (pp. 1-8).

Popper, K. (1959). *The logic of scientific discovery.* . UK: Basic Books. .

Rehder, B., & Hoffman, A. (2005). Eyetracking and selective attention in category learning. *Cognitive Psychology, 51*(1), 1-41.

Richman, H. B., Gobet, F., Staszewski, J., & Simon, H. (1996). Perceptual and memory processes in the acquisition of expert performance: The EPAM model. In K. A. Ericsson (Ed.), *The road to excellence: The acquisition of expert performance in the arts and sciences, sports, and games* (pp. 167–187). Mahwah, MA: Erlbaum.

Richman, H. B., & Simon, H. (2002). Simulations of classification learning using EPAM VI. *Complex Information Processing*.

Richman, H. B., Staszewski, J. J., & Simon, H. (1995). Simulation of expert memory using EPAM IV. *Psychological Review, 102*(2), 305-330.

Rosch, E. (1973). On the internal structure of perceptual and semantic categories. In T. E. Moore (Ed.), *Cognitive development and acquisition of language* (pp. 111-144). San Diego: Academic Press.

Rosch, E., Simpson, C., & Miller, R. S. (1976). Structural bases of typicality effects. *Journal of Experimental Psychology: Human Perception and Performance, 2*(4), 491-502. doi:10.1037/0096-1523.2.4.491

Rutledge-Taylor, M., Lebiere, C., Thomson, R., Staszewski, J., & Anderson, J. (2012). A comparison of rule-based versus exemplar-based categorization using the ACT-R architecture. In *Behavior representation in modeling and simulation*. Florida, USA.

Schiller, M., & Gobet, F. (2014). Cognitive models of gambling and problem gambling. In F. Gobet & M. R. G. Schiller (Eds.), *Problem gambling: Cognition, prevention and treatment* (pp. 74-103). London: Palgrave Macmillan.

Shepard, R. N., Hovland, C. L., & Jenkins, H. M. (1961). Learning and memorization of classifications. *Psychological Monographs, 75*(13), 517.

Simon, H. (1974). How big is a chunk? *Science, 183*(4124), 482-488.

Simon, H., & Gilmartin, K. (1973). A simulation of memory for chess positions. *Cognitive Psychology, 5*, 29-46.

Smith, R., Gobet, F., & Lane, P. (2007). An investigation into the effect of ageing on expert memory with CHREST. In *Proceedings of the seventh UK workshop on computational intelligence* (Vol. 2007). Aberdeen.

Scaper: A Static Code Analyzer for Production System Architectures

Hillmer Chona (hac5324@psu.edu)
Frank E. Ritter, (frank.ritter@psu.edu)
College of Information Sciences and Technology
The Pennsylvania State University, University Park, PA 16802 USA

Abstract

We created *Scaper*, a **S**tatic **c**ode **a**nalysis tool for **p**roduction system architectures. Using this tool, we successfully analyzed 26 ACT-R models. This process took approximately 5 seconds to complete and identified 3,619 components, including chunk-types, declarative memories, and productions. In general, the analysis shows appropriate levels of compliance to the Scaper's tests. Models' compliance scores are above 94% for component usage, and above 78% for production design. However, further analysis revealed that 3.9% of the 152 chunks are not used, and 4.1% of the references from the 1,222 productions have discrepancies with the chunk-type definitions. Although we did not analyze whether such issues affect models' performance, their detection at least suggest code improvements opportunities. Moreover, Scaper can support the development of production systems by (a) providing a better visibility of the components, (b) identifying discrepancies in the components' relationships, and (c) fostering models' code readability, maintainability, and scalability.

Keywords: Production System Architectures; ACT-R; Development Tools; Code Static Analysis

Introduction

Static code analysis tools aim to enhance the software development experience, facilitate maintainability, and contribute to the improvement of software quality. We can trace the definition of such tools at least back to the 1970s (Ramamoorthy & Ho, 1975; Wendel & Kleir, 1977). Furthermore, after Johnson (1978) introduced a Unix tool to check C programs, the name of his tool, *lint,* became synonymous with the term for any tool used for source code analysis.

Research have extensively studied these tools (e.g., Bacon & Sweeney, 1996; Louridas, 2006; Mantere et al., 2009), including their capabilities, performance, and benefits. Their proven benefits have inspired both commercial solutions (e.g., FORTINET SAST, HCL AppScan), and open-source projects (e.g., PMD Source Code Analyzer, and SonarQube)[1].

Despite their expertise, all users are susceptible to making errors (Patel et al., 2011). Therefore, detecting errors and user-generated mistakes when creating models can be important (e.g., Ritter et al., 2000). Tools to analyze code usually help to detect potential pitfalls, improve code maintainability, facilitate software scalability, and help to prevent errors. The development of production system-based cognitive architectures will benefit from having tools that give feedback to their developers regarding the model implementation.

Production systems architectures are high-level programming languages (Anderson & Kline, 1977). Their development often presents multiple challenges. For example, the specific characteristics needed to model human cognition (Young, 2001), the steep learning curve (Ricupero, 2024), and the limitations of development tools (Wang, 2024), among others. Moreover, these architectures usually model complex human cognition processes such as trust (Blaha et al., 2020; Lebiere et al., 2021), fatigue (Gunzelmann et al., 2009; Swan et al., 2024), physiology (Dancy, 2013), and problem solving (Ritter & Bibby, 2008). Due to the complexity and these challenges, production systems developers frequently encounter coding errors.

Aiming to foster high-quality code for production systems models and facilitate the development, we have developed *Scaper*, a static code analyzer for ACT-R models (Anderson, 2007; Ritter et al., 2019). *Scaper* can describe components used by ACT-R models and highlight discrepancies in their use. We have also tested this tool by analyzing the ACT-R publication repository (http://act-r.psy.cmu.edu/publication). The tool, the model analysis, and potential impacts are presented in the following sections.

The Static Code Analyzer

Scaper parses the model's components, identifies potential issues in the relationships between its components, and estimates the model's compliance with ACT-R syntax. This tool evaluates the models using syntax rules defined

[1]An extensive list is available at the National Institute of Standards and Technology (NIST) website https://www.nist.gov/itl/ssd/software-quality-group/source-code-security-analyzers.

in the ACT-R 7.28+ Reference Manual (http://act-r.psy.cmu.edu/actr7.x/reference-manual.pdf) and an internal ruleset we created. Although, We built Scaper as a Java program [2]; it runs as a native standalone app. It could also be extended to function as a mode in Emacs or an extension for Eclipse. Scaper can analyze multiple models in batch and generates reports in text files. Figure 1 shows an example of a simple model.

```
(define-model a-model
(chunk-type number number next)
(chunk-type add arg1 arg2 sum count)
(add-dm
 (zero ISA number number zero next one)
 (one ISA number number one next two)
 (test-goal ISA add arg1 five arg2 two))
(P init-add
   =goal>
      ISA          add
      arg1         =num1
      arg2         =num2
      sum          nil
  ==>
   =goal>
      ISA          add
      sum          =num1
      count        zero
   +retrieval>
      ISA          number
      number       =num1
   =log>
      ISA          log
 )
)
```

Figure 1: An ACT-R model that uses two chunk-types: *number* and *add*; three declarative memories: *zero*, *one*, and *test-goal*; and a production: *init-add*.

Model files serve as the input to the analysis process. Figure 2 shows the analysis workflow. This analysis follows three sequential steps: reading, mapping, and analysis. In the first step, the reader filters according two criteria: the use of the *define-model* clause and the type of components defined within the *define-model* block. Only models that define chunk-types, declarative memories, and/or productions within the *define-model* clause proceed to the mapping step.

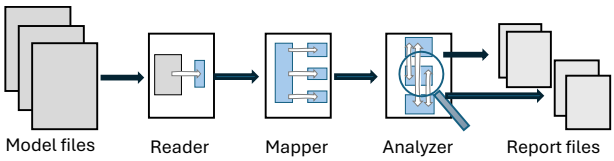


Figure 2: Analysis workflow.

In the second step, the mapper decomposes every component into the elements defined for its syntax. Finally, in the third step, the analyzer scans every component to (a) find usages and dependencies, and (b) detect potential discrepancies in their relationships.

The output of the analysis consists of two reports: Component usages, and Production design, each of them with summary and detail versions. These four reports are generated in text file format. While the summary reports provide a high-level overview of the model's compliance, the detailed version presents the evaluation of each component.

Summary Reports

The summary reports are similar in that they consolidate the findings for both the model and its components but differ in their scope. Figures 3 and 4 show the summary reports for the sample model in the Figure 1.

Figure 3 shows the component usage report for the sample model. This report focuses on components such as chunks-types, declarative memory, and productions. On the other hand, the production design report, shown in Figure 4, covers the production structure, especially its relationship with the chunk-types and their slots of information.

```
_____________________________________________________________
Model: a-model
Model-File: a-model.lisp
Compliance: 100%
Summary_____________________________________________________
Chunk-Types: 2
Declarative Memories: 3
Productions: 1
Details_____________________________________________________
* Chunk type "number" Compliance: 100%      Used: 3 times
 Used by "zero" (Declarative memory)
 Used by "one" (Declarative memory)
 Used by "init-add" (Production)
_____________________________________________________________
```

Figure 3: An example of the Component usages summary report.

The summary reports also include a compliance score for the model. These scores are given in percentages. For component usages the score corresponds to the percentage of components that are in use. For the production design compliance score, the analyzer applies penalties for the following issues (a) slots that do not match the chunk-type definition (5 points), (b) chunks that do not follow a chunk-type definition, but implies a chunk definition that can be inferred and checked against other productions (10 points), we call these chunks *implicit chunks*, and (c) chunks without slots (50 points).

We assign a high penalty of 50 points to chunks without slots because slots are essential part of chunks; a 10 point penalty to the usage of implicit chunks, because although chunk-types are optional, describing them increases code readability; and finally 5 points, the lowest penalty, reflects the good practice of using a chunk-type

[2]Code available at https://github.com/psu-acs/scaper

but penalize the use of slots that are not part of the chunk-type definition. These penalties do not imply errors, but they may be roughly equivalent to warning and errors in compliers. Furthermore, we use these penalty mechanisms to highlight the components that production system developers should review in depth.

```
Model: a-model
Model-File: a-model.lisp
Compliance: 90.7%
Summary
Explicit Chunk-Types: 3
Implicit Chunk-Types: 1
Matched Slots: 6
Unmatched Slots: 0
Details
* Production "init-add" Compliance: 90.7%
  Buffers in conditions: 1
  Buffers in action: 2
```

Figure 4: An example of Production design summary report.

Detailed Reports

Scaper generates two detailed reports: Component usage and Production design. These reports contain all the specifics of the analysis. The Component usage report lists the relationship between components and their consumers. For example, Figure 5 shows a Chunk-type named "number" that is used by two declarative memories: "zero" and "one", and by the production "init-ad".

```
Model: a-model
Model-File: a-model.lisp
Compliance: 100%
Detail
Component   Type         Compl.  Consumer  Type (Consumer)
number      Chunk-type   100%    zero      Declarative-memory
number      Chunk-type   100%    one       Declarative-memory
number      Chunk type   100%    init-add  Production
```

Figure 5: An extract from the Component usage report.

The Production design report details the chunks and their designated slots that constitute each production. In this report, each row typically corresponds to a slot. However, for chunks without any slots, the report uses the keyword "*null*" to represent missing slots. Figure 6 shows an extract form a Production design report. This extract shows the production "init-add" with 6 slots (arg1, arg2, sum, sum, count, number), and the chunk "log" without any slots, represented by the keyword *null*.

In addition, to the four reports, Scaper also generates log files. These files contain the detail of the processing component by component, as well as the blockers that prevents the completion of the analysis for a given model, (e.g., a file missing the define-model clause). The processing logs are not part of the scope of our analysis.

```
Model: a-model
Model-File: a-model.lisp
Compliance: 90.7%
Detail
Production Scope      Buffer-Name Chunk   Slot   Compl.
init-add   Condition  =goal>      add      arg1    100%
init-add   Condition  =goal>      add      arg2    100%
init-add   Condition  =goal>      add      sum     100%
init-add   Action     =goal>      add      sum     100%
init-add   Action     =goal>      add      count   100%
init-add   Action     +retrieval> number   number  100%
init-add   Action     =log>       log      null     35%
```

Figure 6: An extract from the Production design report.

Application

We tested Scaper by scanning the ACT-R models repository. We found 46 publications reporting models' code attachments. However, 14 of those publications do not include model files. This situation is similar to the one reported by Thimbleby (2003) who found that about a 30% of papers with model-based experiments in the *Journal of Machine Learning Research* did not make their code available.

After downloading and uncompressing the 46 publications' attachments, we detected 155 text files that contain source code structures (some publications contain more than one model file). We setup a MacOS machine with 8GB of RAM, and a 3.1 GHz Dual-Core Intel Core i5 processor with both the analyzer and the files to run the analysis.

The analysis took about 5 seconds to complete: 2.8 seconds for the component usage report, and 1.7 seconds for the production memory design. That is to say, this analysis does not require extensive processing. Scaper successfully completed the analysis of 26 files reported in 21 publications. The analysis of the remaining 129 files is incomplete because they do not contain any definitions of the ACT-R's components that Scaper checks. Some of those files use the ACT-R versions available before the introduction of the clause "define-model" (e.g., versions 5 and 5.1), and others contains only functions, macros and other utility components. Therefore, we report and discuss only the models with the components found in the 26 files.

Results and Discussion

In general, the static code analysis demonstrates a good level of compliance regarding Scaper's validation rules. However, the code analysis uncovered potential issues that could indicate underlying problems. We consolidated the analysis across all these models to gain insights into the models' code quality.

Component Usages

In the 26 models' files, we found an average of ~109.5 components per model file, distributed as follows: 152 explicit chunk-types, 1,484 declarative memories, and 1,212 productions. We observed a high usage of the chunk-types structures, with 96% being in use. These structures are used in a proportion of 69.8% and 30.2% by declarative memories and productions, respectively. Figure 7 shows the frequency of uses chunk-types in chunk definitions.

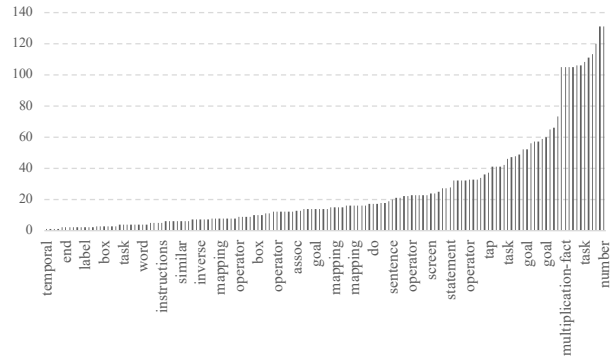


Figure 7: Reuse of chunk-types. The x-axis represents the chunk types of each model, while the y-axis the frequencies. Some of these chunk-types appear more than once (e.g., goal, box, and task), because different models use the same names to define their chunks.

We observed also a positive trend in reusing chunk-types structures within models, with a median of 14 times. On average, chunk-types are reused 25.8 times. The names of the chunk types are tied to the models being analyzed. But the names show that some architecture-related chunks get used often (e.g., goal) and that some task-related chunks (e.g., number) are also used often.

Another relevant finding is the usage of 771 implicit chunks. Unlike explicit chunks, implicit chunks do not use a chunk-type definition. These chunks are typically defined within the productions. The absence of a chunk-type definition is not an error, as the use of chunk-types is optional in the latest ACT-R versions. However, using chunk-types can enhance the model's code readability and maintainability.

Production Memory Design

A production typically encompasses buffers contained in conditions or actions. These buffers have access to chunks and react to their changes (Newell, 1973). Scaper identified 1,212 productions, an average 46.6 per model. Figure 8 shows the number of productions for the 26 models.

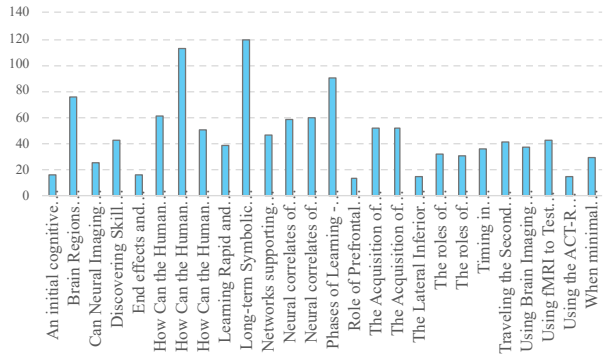


Figure 8: Number of productions by model. Some publications contain multiple models (e.g., "How Can the Human…"). Therefore, each model corresponds to a separate bar.

In total, these productions use 7,442 slots across 3,187 chunks. 82.4% of those chunks follow an explicit chunk-type definition, and 17.6% use an implicit chunk-type definition. Regarding the slots, 6,249 matched the slots defined in the explicit chunk-types, 270 do not match them, and 923 correspond to slots defined in implicit chunks. The analysis also reveals that 382 chunks initialized in productions do not use any slot, even when 45.7% of those chunks are explicitly defined. Figure 9 shows the number of slots used across all the productions of each of the 26 models.

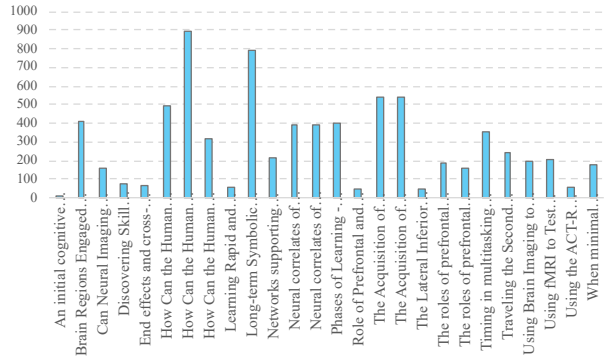


Figure 9: Number of slots used by model. Some publications contain multiple models (e.g., "How Can the Human…"). Therefore, each model corresponds to a separate bar.

Models Compliance

Figure 10 shows the models' compliance scores. In general, the analysis reveals good compliance, with models achieving over 94% for all component usages, and above 78% for productions design, and with a median of 91.6% for production design.

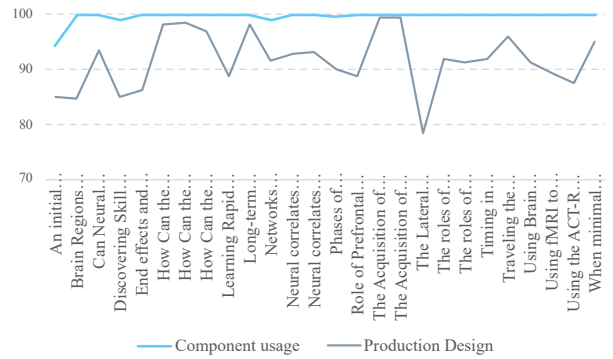


Figure 10: Models compliance evaluation. Some publications contain multiple models (e.g., "How Can the Human…"). Therefore, the models' compliance is plotted separately for each model.

Potential issues

Because the models come from published studies, the detected issues may not compromise the models' output. They may reflect dead (unused) code or code for debugging. Whether those issues affect the models' predictions are out of the scope of our study. Moreover, we cannot yet categorize these issues as errors. Table 1 summarizes the issues that Scaper detected in this set of models.

There were 6 instances where a chunk-type was defined, but the chunk was not used. Inspection of the models suggests that these might be models with an unused chunk for capabilities that were never created. The other issues mainly impact code readability, making it difficult to evolve and reuse the models.

Table 1: Detected Issues

Issues	Frequency
Unused chunk-types	6
Implicit chunks	581
Chunks without slots filled	382
Unmatched slot definitions	270
Implicit slots used in productions	923

Discussion

Our primary goal was to create a tool to gain insights into the code of production system architectures and potentially provide a way to support model creation. We developed Scaper, a static code analyzer for ACT-R models, and used it to analyze 26 cognitive models. The analysis demonstrated the effectiveness of Scaper.

Furthermore, Scaper allowed us to characterize cognitive models published over a period of 20 years. This characterization showed that the ACT-R community could benefit from incorporating further software development and engineering practices to foster our research (e.g., Wilson et al., 2017).

The adoption of tools such as Scaper has the potential to significantly impact the evolution of other production systems development as well, such as Soar (Laird, 2017) or other production systems architectures (Kotseruba & Tsotsos, 2022; in press). In particular, tools like Scaper can provide succinct visibility into models' design, reduce modelers' learning curves, support the early stages of production system development, and help identify opportunities to improve and simplify model code.

Limitations

Despite its demonstrated effectiveness, we recognize that Scaper has several limitations worth noting. First, it uses a limited ruleset for static analysis. Because ACT-R has several components and production systems, and developers use different programming techniques, there are structures that Scaper does not yet understand. To address this gap, we will continue the technical evolution of our tool.

Second, it is worth pointing out that Scaper lacks external validation of its own usability. Both development tests and this study were conducted only by us. We plan to conduct a study to gather feedback from potential users. This feedback will help us not only validate the performance of Scaper, but also the enrich its capabilities.

Another limitation is that the current version runs as a standalone process. Further development is required to integrate Scaper with other tools (e.g., Emacs, Eclipse). This integration will offer capabilities such as real-time validation, and a unified development experience.

Finally, Scaper has only a detector profile. Although detecting issues facilitates development, users would benefit from having explanations and guidance. We plan to improve the Scaper's output by providing details of issues and offering recommendations for improving the code.

Conclusion

We have presented Scaper, which, to the best of our knowledge, is the first Static Code Analyzer for the ACT-R production system architecture. We demonstrated its utility by analyzing 26 published cognitive models. This analysis showed that Scaper can effectively detect code issues and, therefore, contribute to creating high-quality code for science (Thimbleby, 2024). We hope that our tool serves as a stepping stone for the development of production systems tools that will facilitate, improve, and enhance the creation of cognitive and agent architectures.

References

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

https://doi.org/10.1093/acprof:oso/9780195324259.001.0001

Anderson, J. R., & Kline, P. J. (1977). Design of a production system for cognitive modeling. *ACM SIGART Bulletin*, *63*, 60–65. https://doi.org/10.1145/1045343.1045377

Bacon, D. F., & Sweeney, P. F. (1996). Fast static analysis of C++ virtual function calls. *Proceedings of the 11th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 324–341. https://doi.org/10.1145/236337.236371

Blaha, L. M., Lebiere, C., Fallon, C. K., & Jefferson, B. A. (2020). Cognitive mechanisms for calibrating trust and reliance on automation. *Proceedings of the 18th International Conference on Cognitive Modelling (ICCM 2020)*.

Dancy, C. L. (2013). ACT-RΦ: A cognitive architecture with physiology and affect. *Biologically Inspired Cognitive Architectures*, *6*, 40–45. https://doi.org/10.1016/j.bica.2013.07.008

Gunzelmann, G., Gross, J. B., Gluck, K. A., & Dinges, D. F. (2009). Sleep deprivation and sustained attention performance: Integrating mathematical and cognitive modeling. *Cognitive Science*, *33*(5), 880–910. https://doi.org/10.1111/j.1551-6709.2009.01032.x

Johnson, S. C. (1978). *Lint, a C program checker*. Bell Laboratories.

Lebiere, C., Blaha, L. M., Fallon, C. K., & Jefferson, B. (2021). Adaptive cognitive mechanisms to maintain calibrated trust and reliance in automation. *Frontiers in Robotics and AI*, *8*, 652776. https://doi.org/10.3389/frobt.2021.652776

Louridas, P. (2006). Static code analysis. *IEEE Software*, *23*(4), 58–61. IEEE Software. https://doi.org/10.1109/MS.2006.114

Mantere, M., Uusitalo, I., & Roning, J. (2009). Comparison of static code analysis tools. *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, 15–22. https://doi.org/10.1109/SECURWARE.2009.10

Newell, A. (1973). Production systems: Models of control structures. In *Visual Information Processing* (pp. 463–526). Elsevier. https://doi.org/10.1016/B978-0-12-170150-5.50016-0

Patel, V. L., Cohen, T., Murarka, T., Olsen, J., Kagita, S., Myneni, S., Buchman, T., & Ghaemmaghami, V. (2011). Recovery at the edge of error: Debunking the myth of the infallible expert. *Journal of Biomedical Informatics*, *44*(3), 413–424. https://doi.org/10.1016/j.jbi.2010.09.005

Ramamoorthy, C. V., & Ho, S. F. (1975). Testing large software with automated software evaluation systems. *Proceedings of the International Conference on Reliable Software*, 382–394. https://doi.org/10.1145/800027.808461

Ricupero, S. (2024). *Modeling working memory: Varying noise based on neural differences* [Doctoral dissertation, The Pennsylvania State University].

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. https://doi.org/10.1145/353485.353486

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2019). ACT-R: A cognitive architecture for modeling cognition. *WIREs Cognitive Science*, *10*(3), e1488. https://doi.org/10.1002/wcs.1488

Swan, G., Stevens, C. A., Veksler, B. Z., & Morris, M. B. (2024). Modeling fatigue in the N-back task with ACT-R and the fatigue module. *Proceedings of the 22nd International Conference on Cognitive Modelling (ICCM 2024)*.

Thimbleby, H. (2003). Give your computer's IQ a boost, review of Journal of Machine Learning Research. *The Times Higher Education Supplement*, *1588*, 26.

Thimbleby, H. (2024). Improving science that uses code. *The Computer Journal*, *67*(4), 1381–1404. https://doi.org/10.1093/comjnl/bxad067

Wang, S. (2024). *Understanding pain points of ACT-R modelers: A human-centered approach* [Doctoral dissertation, The Pennsylvania State University].

Wendel, Irv. K., & Kleir, R. L. (1977). FORTRAN error detection through static analysis. *ACM SIGSOFT Software Engineering Notes*, *2*(3), 22–28. https://doi.org/10.1145/1012319.1012323

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLoS Computational Biology*, *13*(6), e1005510. https://doi.org/10.1371/journal.pcbi.1005510

Young, R. M. (2001). Production systems in cognitive psychology. In N. J. Smelser & P. B. Baltes (Eds.), *International Encyclopedia of the Social & Behavioral Sciences* (pp. 12143–12146). Pergamon. https://doi.org/10.1016/B0-08-043076-7/01597-7

Metacognitive Mechanisms of the Attentional Training Technique

Brendan Conway-Smith (brendan.conwaysmith@carleton.ca),
Robert L. West (robert.west@carleton.ca)
Department of Cognitive Science, Carleton University, Ottawa, ON K1S 5B6 Canada

Abstract

The Attentional Training Technique (ATT) has been shown to enhance attentional control and reduce maladaptive cognitive patterns but lacks a well-defined computational explanation. This paper applies a metacognitive skill model within the ACT-R cognitive architecture to clarify the procedural mechanisms underlying ATT. Grounded in proceduralization theory, we propose that ATT transforms declarative attentional strategies into automatic procedural skills, enhancing metacognitive control and emotional regulation. This framework advances our understanding of the computational and cognitive mechanisms supporting ATT, its applications in psychotherapy, and the process of metacognitive skill learning.

Keywords: Attentional training technique; attention; metacognition; metacognitive skill; ACT-R; meditation

Introduction

Understanding the cognitive mechanisms underlying psychotherapeutic interventions is crucial for optimizing their efficacy and refining treatment strategies. The Attentional Training Technique (ATT) (Wells, 1990; 2019) has been shown to be effective in alleviating symptoms across various psychological disorders (Rochat, Manolov & Billieux, 2018). However, the computational and cognitive mechanisms that give rise to its effectiveness remain poorly understood. This paper investigates these mechanisms using the ACT-R cognitive architecture to provide a more precise account of these processes.

In their 2023 fMRI study, Jahn et al. stated, "Understanding the 'how' behind the Attentional Training Technique should lead to a better understanding of attentional control and metacognition in general and could eventually manifest in improved or even more specific treatment" (p.12). To this end, we aim to articulate the mechanisms of attentional training by applying a model of metacognitive skill that has been effectively used to explore related cognitive processes, including metacognitive sensitivity, emotional regulation, and attentional control (Conway-Smith, West, & Mylopoulos, 2023). Grounded in the principles of proceduralization, this model provides a computational framework for understanding how metacognitive skills develop and refine over time, offering a mechanistic account of how training enhances metacognitive monitoring and control.

This paper will help address Wells' (2019) call for a "stronger information processing theory" (p.13) to explain metacognitive control — its components, functions, and the types of metacognitive information involved in the preservation and disengagement of negative processing. A more precise theoretical account of the Attentional Training Technique's subcomponents may not only enhance its existing applications but also facilitate the development of more effective interventions.

This study aims to address unanswered questions: What cognitive mechanisms underpin attentional control training? How does their enhancement reduce maladaptive patterns of thought and emotion? More broadly, we explore how intelligent systems may enhance their ability to monitor and regulate their own activity.

To address these questions, we first provide an overview of the Attentional Training Technique and its practical applications. Next, we outline key aspects of metacognition and the metacognitive skill model. We then apply this model to ATT to illuminate its constitutive mechanisms. Finally, we discuss how this refined explanation enhances our understanding of the cognitive processes that support emotional regulation and alleviate psychological symptoms.

Attentional Training Technique

Psychotherapeutic treatments in metacognitive therapy are grounded in the Self-Regulatory Executive Function (S-REF) model, which explains the role of strategic processes and metacognition in psychological disorders (Wells & Matthews, 1996). The S-REF model posits that maladaptive metacognitive beliefs and knowledge can trigger an adverse thought pattern known as the Cognitive Attentional Syndrome (CAS). CAS is a style of negative processing characterized by worry, rumination, and threat monitoring. It involves rigid, self-focused attention that amplifies negative emotions, leading to persistent self-preoccupation and distress. CAS is also associated with maladaptive coping strategies such as thought suppression, avoidance behaviors, and substance abuse (Wells, 2009).

The Attentional Training Technique (ATT) is designed to counteract CAS by enhancing metacognitive control and breaking cycles of negative thought (Knowles & Wells, 2018). ATT helps individuals disengage from persistent thought patterns,

interrupt self-focused attention, and strengthen metacognitive awareness (Knowles et al., 2016; Nassif & Wells, 2014).

fMRI studies (Jahn et al., 2023) have linked ATT to improvements in attentional abilities and structural changes in the brain. However, researchers emphasize that the underlying cognitive mechanisms of ATT remain poorly understood, highlighting the need for a more detailed theoretical framework. This aligns with Wells' (2019) assertion that "a more detailed modeling of the metacognitive and cognitive architectures supporting self-regulatory processing is needed to advance the field" (p. 5).

Metacognition

We propose that the Attention Training Technique fundamentally relies on a form of automatized metacognition. The common conception of metacognition pertains to the monitoring and control of cognitive processes (Flavell 1979; Fleming, Dolan, & Frith, 2012). Metacognitive skill presupposes that the main components of metacognition, monitoring and control, can improve with practice. *Metacognitive control* concerns the active regulation of cognitive processes or states to either activate or suppress them (Proust, 2013; Wells, 2019). The control of one's own cognitive activity can involve a range of processes such as attention, emotion, planning, reasoning, and memory (Slagter et al., 2011; Efklides, Schwartz, & Brown, 2017; Pearman et al., 2020). *Metacognitive monitoring* refers to the ability to recognize and identify cognitive states. It involves the perception of internal mental properties such as thoughts and feelings in order to regulate those states or direct behavior.

There are at least two types of cognitive representations that engage in metacognitive monitoring and control processes — declarative knowledge and procedural knowledge. Metacognitive knowledge, or meta-knowledge, is considered a form of declarative knowledge (Schraw & Moshman, 1995; McCormick, 2003). Meta-knowledge takes the form of an explicit metarepresentation that is propositionally formatted and refers to a cognitive property, e.g.: "I am focused" (Shea et al., 2014; Proust, 2013). Metacognitive knowledge is considered distinct from metacognitive skill, as it does not automatically deploy metacognitive processes (Veenman & Elshout, 1999). Meta-knowledge is further distinguished from a metacognitive instruction, which specifies the mental action to be performed (Wells, 2019). A metacognitive instruction, or meta-instruction, prescribes a mental action directed toward controlling some cognitive process, e.g.: "Direct focus toward the present task."

The executing of metacognitive instruction is performed by way of procedural knowledge. Improvements in metacognition are said to involve the refining of procedural knowledge that people employ to monitor and control their own cognitive processes (Brown & DeLoache, 1978; Schraw & Moshman, 1995; Wells, 2019). The various realms of metacognitive skills can be understood as different domains of procedural knowledge (Veenman et al., 2005; Braithwaite, & Sprague, 2021). The dynamic role of declarative knowledge and procedural knowledge in metacognitive processes can be more clearly articulated within the ACT-R cognitive architecture.

ACT-R

Theories of metacognition have been modeled in the ACT-R cognitive architecture (Reitter, 2010; Anderson & Fincham, 2014). ACT-R instantiates decades of research on the computational mechanisms of human cognition. Its mandate is to depict the components necessary for human intelligence, which include working memory, perception, action, declarative memory, and procedural memory. These modules have been correlated with their associated brain regions, providing a neurobiologically grounded framework for investigating cognitive processes (Borst et al., 2015).

ACT-R distinguishes between declarative knowledge and procedural knowledge to explain the underlying components of skill learning, which accords with the literature on skill in philosophy and psychology (Squire, 1992; Christensen, Sutton, & McIlwain, 2016). Declarative knowledge is formatted propositionally and structured within semantic networks. Procedural knowledge is specified computationally as "production rules" which are a dominant form of representation within accounts of skill (Newell, 1990; Taatgen & Lee, 2003). Production rules, or "productions", transform information and change the state of the system to complete a task or resolve a problem. The building and refining of production rules are considered to be central to human intelligence and fundamental to cognitive skills (Anderson, 1993). Neurologically, production rules are associated with the 50ms decision timing in the basal ganglia (Stocco, 2018).

A production rule is modeled after a computer program instruction in the form of a "condition-action" pairing (Figure 1). It specifies a condition that, when met, performs a prescribed action. A production can also be thought of as an "if-then" rule. *If* the conditional side matches to a pattern in working memory, *then* it fires a prescribed action (Anderson, 1993; Stocco et al., 2021).

Figure 1: Production rules are formatted as an if-then rule, or condition-action pairing. *If* the condition side matches to the cue in working memory, *then* it fires an action.

This is clarified by noting that procedural knowledge (production rules) is generally not innate in humans. For example, a child must develop production rules to print their name (motor actions), perform mathematical calculations (cognitive actions), and regulate their focus (metacognitive actions). They must learn that conditions such as 'print name,' 'solve for x,' or 'pay attention' are paired with the appropriate action sequences. Once these actions are associated with the correct cues, practice is required to refine the supporting production rules and improve performance.

With sufficient practice, these productions become stored in procedural memory. When a relevant cue appears in working memory ('print,' 'calculate,' 'focus'), matching productions will activate and execute the correct actions. In this way, cues in working memory can trigger procedural knowledge across motor, cognitive, and metacognitive domains, and refined through a process of proceduralization.

Proceduralization

Proceduralization is a key concept in skill acquisition, describing the transition from explicit declarative knowledge to implicit procedural knowledge. Theories of skill learning characterize this process as moving from a declarative stage of rule-following to a procedural stage where performance becomes faster, more automatic, and more accurate (Dreyfus & Dreyfus, 1986; Kim & Ritter, 2015). Our account follows Fitts' (1964) skill acquisition model as computationally interpreted by Anderson (1982).

Proceduralization plays a central role in both physical skills, such as those in athletics (Beilock & Carr, 2001; Ford, Hodges & Williams, 2005), and cognitive skills, such as mathematics (Anderson, 1982; Taatgen & Lee, 2003). As declarative knowledge is retrieved and practiced, actions become faster, more automatic, and less error-prone. This occurs because procedural knowledge becomes directly associated with task-relevant cues, reducing reliance on slow declarative knowledge retrieval. Consequently, performance speeds up and working memory load decreases. Task refinement can also occur through mechanisms such as time-delayed learning, where faster productions are reinforced.

With sufficient practice, conscious control diminishes, and skill execution becomes automatized. Automatized skills operate largely outside of working memory, making them less perceivable to the performer (Beilock & Carr, 2004; Ford et al., 2005). When an appropriate cue appears, well-practiced motor, cognitive, and metacognitive skills activate automatically with minimal effort. We posit that this transition toward automaticity is a fundamental mechanism underlying the Attentional Training Technique.

Stages of metacognitive skill

Here we propose that proceduralization is key to understanding attentional training as a subdomain of metacognitive skill. Metacognitive proceduralization articulates a top-down mechanism by which human cognition becomes more skillful at monitoring and controlling its own processes, such as attention, emotion, and metacognitive sensitivity (Conway-Smith, West, & Mylopoulos, 2023; Conway-Smith & West, 2024). Explanations of metacognitive skill have also produced bottom-up models, where implicit processes learn by way of stored low-level feedback (Proust, 2013) such as metacognitive reinforcement learning (Krueger, Lieder & Griffiths, 2017).

Metacognitive proceduralization posits that metacognitive skill develops from an initial stage of instruction-following to an advanced stage where performance largely relies on automatic procedural knowledge (production rules). In the later stages, monitoring and control processes are deployed quickly, more automatically, and require less working memory. This shift towards automatization not only enhances the efficiency of cognitive processes but also frees up cognitive resources, allowing for more complex and nuanced metacognitive operations.

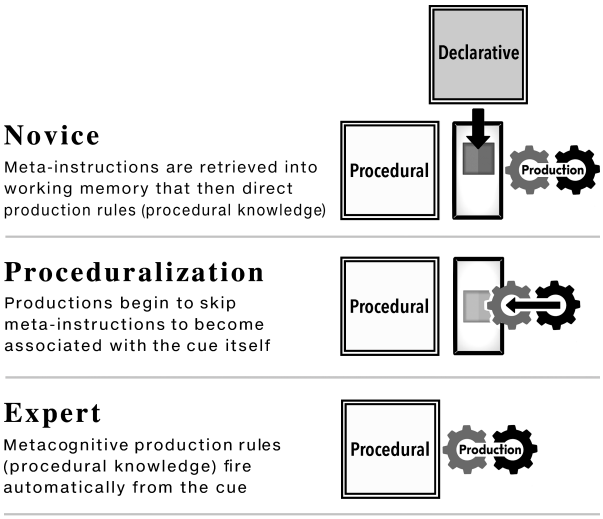


Figure 2: Three stages of metacognitive skill learning through proceduralization (Conway-Smith, West, & Mylopoulos, 2023).

According to the theory of metacognitive learning via proceduralization, a metacognitive practitioner progresses through three stages of training (Figure 2), which we propose as a structured framework for understanding the Attentional Training Technique.

Novice Stage. Training begins with meta-instructions that direct monitoring and control processes toward a specific target of focus (e.g., a visual point, physical area, or sound). These instructions are retrieved from declarative memory and executed as productions. Performance at this stage is slow, effortful, error-prone, and demands significant working memory resources.

Intermediate Stage. Through proceduralization, repeated practice refines meta-instructions into faster, task-specific production rules, reducing reliance on declarative knowledge. These specialized productions are rewarded and reinforced, allowing metacognitive performance to become quicker, more automatic, and less cognitively demanding.

Expert Stage. Meta-instructions are fully converted into procedural knowledge and stored in memory. Upon encountering a relevant stimulus (e.g., a cue to focus attention), production rules activate automatically, requiring minimal conscious effort. At this stage, metacognitive performance is fast, efficient, and highly automatic, demonstrating the hallmarks of expertise.

Empirical support for proceduralized attention

Empirical findings support the notion that attentional control can become proceduralized through training. For example, Ramamurthy and Blaser (2017) introduced the concept of "procedural attention" to describe the transition from deliberate to automatic attentional deployment. In their study, participants were instructed where and how to allocate attention, and with repeated practice, attention became automatically oriented toward those rehearsed locations. This was interpreted as evidence for an "offline" attentional selection mode, that is, cognitively unsupervised and automatic. The authors noted that this mode is "analogous to the procedural memory that guides skilled motor behavior" (p. 1).

Additional evidence for proceduralization comes from data consistent with the power law of skill acquisition. Logan (1988) operationalized automatization as a speed-up in reaction times (RTs) that follows a power function — characterized by rapid initial improvement followed by a gradual leveling off. This negatively accelerating learning curve has been widely observed in both motor and cognitive skill domains (Newell & Rosenbloom, 1981; Anderson, 1982).

Shin et al. (2015) provide evidence that attentional control improves with practice according to a power law. In a multi-session rapid serial visual presentation (RSVP) task, participants showed gains in target identification and reductions in attentional blink, both following a negatively accelerating curve typical of procedural skill learning. These findings suggest that attentional control, like other motor and cognitive skills, develops through structured practice and proceduralization.

Clarifying the Attentional Training Technique

Building on the theory of metacognitive learning via proceduralization, we apply this framework to identify the key features and stages of proceduralization in the Attentional Training Technique (ATT). This structured, computational approach characterizes how attentional control transitions from deliberate, declarative strategies to automatic processes.

Jahn et al. (2023) describe the ATT method as implemented via a standardized audio protocol based on the Metacognitive Therapy (MCT) manual (Wells, 2009). Participants receive instructions on directing their attention while listening to six simultaneous audio stimuli: a bell, traffic noise, birds, rushing water, crickets, and a ticking clock. Each 12-minute session consists of three phases: selective attention (focusing on one sound at a time), attentional switching (shifting between sounds), and divided attention (attending to multiple sounds simultaneously).

Participants practiced ATT twice daily for five days. By the study's conclusion, they exhibited improved metacognitive expertise, demonstrating faster and more accurate attentional control compared to a control group. This training trajectory mirrors the transition from effortful, declarative instruction-following to automatized proficiency — a hallmark of metacognitive proceduralization.

Jahn et al. (2023) suggest that the anterior cingulate cortex (ACC) plays a primary role in metacognitive development and the storage/execution of procedural knowledge for attentional control. While ACT-R traditionally attributes procedural execution to the basal ganglia, both regions likely contribute within a broader neural network governing metacognitive proceduralization. Here, we propose the ACC and basal ganglia operate in tandem, collectively supporting the automatization of attentional control processes in ATT.

Alternative explanations of attentional training have emphasized mechanisms such as reinforcement learning (Krueger et al., 2017) and predictive coding (Clark, 2015). While these accounts provide valuable perspectives, they do not fully explain the transition from controlled to automatic attentional regulation. Proceduralization uniquely captures how attentional skills become automatized through training, offering a more mechanistic account of skill acquisition in ATT.

Exiting maladaptive thoughts

Metacognitive proceduralization provides a framework for understanding how attentional control training mitigates symptoms of psychological disorders. The Self-Regulatory Executive Function (S-REF) model describes how disorders such as anxiety and depression involve perseverative negative thinking (e.g., worry, rumination), characterized by repetitive cognitive loops (Wells, 1995, 2000). Cognitive Attentional Syndrome

(CAS) exemplifies this process, where threat-related thoughts become self-reinforcing without a natural exit condition from the loop.

A computational perspective helps clarify how attentional training disrupts maladaptive cognitive cycles. In ACT-R, production rules (procedural knowledge) operate as condition-action pairs. If a condition in working memory is met, an action is executed. Through attentional training, production rules become more efficient at recognizing maladaptive thought patterns, facilitating disengagement.

This aligns with clinical insights on the importance of developing meta-awareness, or "identifying thoughts as thoughts" (Moore, 1996) — a crucial step in breaking repetitive negative thinking. For example, an individual prone to rumination may, through attentional training, develop automatized metacognitive productions that detect and disengage from intrusive thoughts, effectively providing an exit condition (Figure 3).

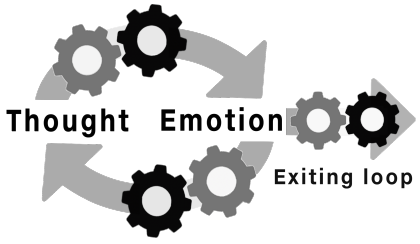


Figure 3: Maladaptive thought and emotional loops can persist without an exit condition. Attentional training develops production rules that recognize and disengage from these patterns, thereby exiting the loop.

Exiting negative emotions

Attentional training has been shown to reduce negative emotional patterns, yet the precise underlying mechanism remains unclear (Wadlinger & Isaacowitz, 2011; Wells, 2019). We propose that the same procedural mechanism that enables production rules to detect and disengage from maladaptive thoughts can also be applied to negative emotions.

This claim is supported by evidence that both declarative knowledge (propositional information) and emotions (non-propositional affective states) are represented as patterns of information within working memory and are accessible to production rules (West & Conway-Smith, 2019). As a result, attentional training fosters the development of production rules that recognize and disengage from negative emotional states, mirroring the mechanism by which it disrupts maladaptive thoughts. This process aligns with previous research suggesting that emotional regulation is supported by metacognitive proceduralization (Conway-Smith & West, 2024).

From a computational perspective, an exit condition from any cognitive loop is implemented through an exit production — a procedural rule that activates upon detecting a maladaptive emotional state. This shared mechanism suggests that metacognitive control over thoughts and emotions relies on a common process of proceduralization. As a result, attentional training provides an integrated framework for cognitive and emotional regulation, offering a mechanistic account of how structured interventions disrupt cycles of maladaptive thought and emotional reactivity.

Theoretical support for transfer effect

This analysis provides theoretical grounding for empirical findings that suggest attentional training skills are transferable — applicable across diverse tasks and cognitive domains (Ducrocq et al., 2016; Chua et al., 2021). Proceduralization helps explain this phenomenon by identifying the metacognitive informational units (i.e., production rules) that enable attentional skills to be generalized across contexts.

The role of production rules in skill transfer has been extensively studied, demonstrating their ability to facilitate both near and far transfer across cognitive domains (Singley & Anderson, 1989; Taatgen, 2013). From this perspective, attentional skills transfer when production rules become refined and automatized, allowing them to be triggered effortlessly in any context requiring attentional control. Once proceduralized, these skills stabilize focus across various tasks without requiring explicit instruction or deliberate cognitive effort.

Beyond attentional training, this framework suggests that metacognitive skill acquisition follows a generalizable learning trajectory, in which proceduralized attentional control becomes a core cognitive resource that can be redeployed across different domains, ranging from problem-solving and decision-making to emotional regulation and self-directed learning. This underscores the broader cognitive impact of attentional training beyond its immediate therapeutic applications.

Future Directions and Implications

Further empirical work is needed to test and refine this paper's claim that metacognitive proceduralization underlies the Attentional Training Technique (ATT).

First, model validation is critical. Task-based fMRI studies could examine whether attentional proceduralization involves activation in the anterior cingulate cortex and basal ganglia (Jahn et al., 2023). EEG markers may also reveal proceduralization-related changes analogous to those observed in motor and cognitive skill learning, including reduced frontal theta power and diminished prefrontal activation.

Second, the extent to which ATT produces domain-general versus task-specific improvements remains an open question. Clarifying this distinction will be essential for optimizing its application across clinical, educational, and high-performance contexts.

Third, integrating insights from metacognitive reinforcement learning (Krueger, Lieder, & Griffiths, 2017) may enhance the model's ability to represent the dynamics of proceduralization over time.

Fourth, future work could test the hypothesis that behavioral measures of attentional control — such as visual gaze stability, reaction time, and accuracy — follow a power law of learning, with rapid initial gains tapering off with continued practice.

Conclusion

This paper has provided a computational account of the metacognitive mechanisms underlying the Attentional Training Technique (ATT), offering a more precise characterization of how attentional control is proceduralized through structured training. By situating the Attentional Training Technique within the ACT-R cognitive architecture, we extend Wells' (2019) call for a more comprehensive metacognitive information-processing theory, refining our understanding of how metacognitive skills develop and automatize.

Through the application of a metacognitive skill model, we have demonstrated how proceduralization transforms attentional training from an effortful, declarative process into an automatic, self-regulating metacognitive skill. This transition is critical for enhancing attentional control, improving emotional regulation, and disrupting maladaptive cognitive loops characteristic of psychological disorders.

Beyond its clinical implications, this framework suggests that attentional skill training transfers across domains, with potential applications in education and high-performance training. A deeper computational understanding of metacognitive proceduralization can help develop more adaptive, scalable, and personalized interventions for cognitive and emotional self-regulation.

By mapping the metacognitive mechanisms underlying the Attentional Training Technique, we aim to support the development of more effective psychotherapeutic interventions and to advance the broader study of cognitive training, computational modeling, and applied metacognition.

References

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological review*.

Anderson, J. R. (1993). Rules of the Mind. Lawrence Erlbaum Associate, Inc. New Jersey.

Anderson, J. R., Betts, S., Bothell, D., Hope, R., & Lebiere, C. (2019). Learning rapid and precise skills. *Psychological review*

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

Arango-Muñoz, S. (2011). Two levels of metacognition. *Philosophia*.

Beilock, S. L., & Carr, T. H. (2001). On the fragility of skilled performance: What governs choking under pressure?. *Journal of experimental psychology.*

Borst, J. P., Nijboer, M., Taatgen, N. A., van Rijn, H., & Anderson, J. R. (2015). Using data-driven model-brain mappings to constrain formal models of cognition. *PLoS One,* 10(3), e0119673.

Braithwaite, D. W., & Sprague, L. (2021). Conceptual knowledge, procedural knowledge, and metacognition in routine and nonroutine problem solving. *Cognitive Science.*

Brown, A. & DeLoache, J. S. (1978). Skills, plans, and self-regulation. *Children's thinking: What develops.*

Chua, L.-K., Jimenez-Diaz, J., Lewthwaite, R., Kim, T., & Wulf, G. (2021). Superiority of external attentional focus for motor performance and learning: Systematic reviews and meta-analyses. *Psychological Bulletin.*

Christensen, W., Sutton, J., & McIlwain, D. J. (2016). Cognition in skilled action: Meshed control and the varieties of skill experience. Mind & Language.

Clark, A. (2015). *Surfing uncertainty: Prediction, action, and the embodied mind.* Oxford University Press.

Conway-Smith, B., & West, R. L. (2024). The Computational Mechanisms of Detached Mindfulness. In *Proceedings of ICCM 2024, 22nd International Conference on Cognitive Modelling.*

Conway-Smith, B., West, R. L., & Mylopoulos, M. (2023). Metacognitive skill: how it is acquired. In *Proceedings of the Annual Conference of the Cognitive Science Society.*

Dreyfus, H., & Dreyfus, S. (1986). *Mind Over Machine.* New York: The Free Press.

Ducrocq, E., Wilson, M., Vine, S., & Derakshan, N. (2016). Training attentional control improves cognitive and motor task performance. *Journal of sport and exercise psychology.*

Efklides, A., Schwartz, B. L., & Brown, V. (2017). Motivation and affect in self-regulated learning: does metacognition play a role?. In *Handbook of self-regulation of learning and performance.* Routledge.

Fitts, P. M. (1964). Perceptual-motor skill learning. In *Categories of human learning*. Academic Press.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American psychologist.*

Fleming S. M., Dolan R. J., Frith C. D. (2012). Metacognition: Computation, biology and function. Philosophical Transactions of the Royal Society of London, Series B: *Biological Sciences.*

Ford, P., Hodges, N. J., & Williams, A. M. (2005). Online attentional-focus manipulations in a soccer-dribbling task: Implications for the proceduralization of motor skills. *Journal of motor behavior.*

Kim, J. W., & Ritter, F. E. (2015). Learning, forgetting, and relearning for keystroke-and mouse-driven tasks: Relearning is important. *Human–Computer Interaction.*

Knowles, M. M., Foden, P., El-Deredy, W., & Wells, A. (2016). A systematic review of efficacy of the attention training technique in clinical and nonclinical samples. *Journal of clinical psychology.*

Knowles, M. M., & Wells, A. (2018). Single dose of the attention training technique increases resting alpha and beta-oscillations in frontoparietal brain networks. *Frontiers in psychology.*

Krueger, P. M., Lieder, F., & Griffiths, T. (2017). Enhancing metacognitive reinforcement learning using reward structures and feedback. In *Proceedings of the Annual Conference of the Cognitive Science Society.*

Jahn, N., Sinke, C., Kayali, Ö., Krug, S., Leichter, E.,... & Heitland, I. (2023). Neural correlates of the attention training technique as used in metacognitive therapy–A randomized sham-controlled fMRI study in healthy volunteers. *Frontiers in Psychology.*

Newell, A. (1990). *Unified theories of cognition.* Harvard University Press.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 1–55). Hillsdale, NJ: Erlbaum.

Pearman, A., Lustig, E., Hughes, M. & Hertzog, C.(2020). Initial evidence for the efficacy of an everyday memory and metacognitive intervention. *Innovation in Aging.*

Proust, J. (2013). *The philosophy of metacognition: Mental agency and self-awareness.* Oxford.

Proust, J. (2019). From comparative studies to interdisciplinary research on metacognition. *Animal Behavior and Cognition.*

Ramamurthy, M., & Blaser, E. (2017). New rules for visual selection: Isolating procedural attention. *Journal of vision.*

Reitter, D. (2010). Metacognition and multiple strategies in a cognitive model of online control. *Journal of Artificial General Intelligence.*

Rochat, L., Manolov, R., & Billieux, J. (2018). Efficacy of metacognitive therapy in improving mental health: A meta-analysis of single-case studies. *Journal of clinical psychology.*

Schraw, G., & Moshman, D. (1995). Metacognitive theories. *Educational psychology review.*

Shea, N., Boldt, A., Bang, D., Yeung, N., Heyes, C., & Frith, C. D. (2014). Supra-personal cognitive control and metacognition. *Trends in Cognitive Sciences.*

Shin, J. C., Chang, S., & Cho, Y. S. (2015). Adjustment to subtle time constraints and power law learning in rapid serial visual presentation. *Frontiers in Psychology*, 6, 1748.

Singley, M. K., & Anderson, J. R. (1989). The transfer of cognitive skill (No. 9). *Harvard University Press.*

Slagter, H. A., Davidson, R. J., & Lutz, A. (2011). Mental training as a tool in the neuroscientific study of brain and cognitive plasticity. *Frontiers in human neuroscience.*

Squire, L. R. (1992). Declarative and nondeclarative memory: Multiple brain systems supporting learning and memory. *Journal of cognitive neuroscience.*

Stocco, A. (2018). A biologically plausible action selection system for cognitive architectures: Implications of basal ganglia anatomy for learning and decision-making models. *Cognitive science.*

Stocco, A., Sibert, C., Steine-Hanson, Z., Koh, N., Laird, J. E., Lebiere, C. J., & Rosenbloom, P. (2021). Analysis of the human connectome data supports the notion of a "Common Model of Cognition" for human and human-like intelligence across domains. *NeuroImage.*

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human factors.*

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological review*.

Veenman, M., & Elshout, J. J. (1999). Changes in the relation between cognitive and metacognitive skills during the acquisition of expertise. *European journal of psychology of education.*

Wadlinger, H. A., & Isaacowitz, D. M. (2011). Fixing our focus: Training attention to regulate emotion. Personality and social psychology review.

Wells, A. (1990). Panic disorder in association with relaxation induced anxiety: An attentional training approach to treatment. *Behavior therapy.*

Wells, A. (2019). Breaking the cybernetic code: Understanding and treating the human metacognitive control system to enhance mental health. *Frontiers in Psychology.*

Wells, A. (2009). *Metacognitive Therapy for Anxiety and Depression.* New York: Guilford Wells .

Wells, A., & Matthews, G. (1996). Modelling cognition in emotional disorder: The S-REF model. *Behaviour research and therapy.*

Preserving Cognition Through Systems Segregation: A Neural Model of Dedifferentiation and Performance Decline

Spencer Eckler (spencereckler@cmail.carleton.ca)
Nico Turcas (nicoturcas@cmail.carleton.ca)
Mary Alexandria Kelly (MaryKelly4@cunet.carleton.ca)
John A.E. Anderson (johnanderson3@cunet.carleton.ca)
Department of Cognitive Science, Carleton University,
1125 Colonel By Dr, Ottawa, ON K1S 5B6, CA

Abstract

Neural specificity is the ability to differentiate multiple representations in a neural population such that individual neurons provide some constituent feature of those representations (Park, Carp, Hebrank, Park, & Polk, 2010; Kleemeyer et al., 2017). Two different theories of neural specificity have been used to explain interregional brain differences that trend with performance loss: neural dedifferentiation (Koen, Hauck, & Rugg, 2019) and system segregation (Chan, Park, Savalia, Petersen, & Wig, 2014; Wig, 2017). Although both estimates demonstrate a significant trend with performance, neither adequately explains the mechanisms by which performance declines, due to loss in neural specificity. To help identify these mechanisms, we developed three neurocognitive models in the NengoSPA framework that perform a restricted version of a go/no-go task: one control model, and two intervention models. The first intervention was neural population reduction (NPR) in all of one model's network regions, which simulates the process of neural dedifferentiation - the loss of differentiated neural activity. The second intervention was representation localization reduction (RLR) between neural populations in the model's different network regions, which simulates a reduction in system segregation compared to the control model. Performance in both intervention models decreased compared to the control model, although in different ways. The NPR model demonstrated consistent omission errors, while the RLR model demonstrated consistent commission errors. This suggests that a reduction in neural function may cause the omission errors typical of neural attenuation, while compensatory neuroplasticity accounts for the commission errors of neural broadening, due to its limited ability to repair representations damaged by prior neural function loss. Considering these findings, it is likely that estimates of neural specificity are capturing statistically significant reductions in neural function and the inability for neuroplasticity to fully compensate.

Keywords: cognitive model, cognitive architecture, brain simulation, cognitive aging, neural dedifferentiation, system segregation

Introduction

Neural dedifferentiation increases and system segregation decreases with aging in adult populations (Koen et al., 2019; Chan et al., 2014; Wig, 2017). Both phenomena are captured with brain imaging data in which neural activation between brain areas increasingly co-occur with aging. This is coupled with age-related reduction in task performance, suggesting a reduction in neural specificity. Neural specificity refers to the unique activation of neurons in response to distinct stimuli, indicating unique representations and processing for each stimulus (Park et al., 2010; Kleemeyer et al., 2017). Neither neural dedifferentiation nor system segregation fully capture how neural specificity affects the underlying representations. We propose reductions in neural specificity through both a loss of neural function and an increase in compensatory neuroplasticity are the primary causes for the representational decay present in neural dedifferentiation and reductions in system segregation. Additionally, simulated intervention on both neural properties should result in distinct, predictable errors on a cognitive task.

In what follows, we review existing measures of neural dedifferentiation and system segregation and their relationship to task performance. We then present a new computational neurocognitive model of a restricted go/no-go task, developed using the NengoSPA platform (Eliasmith, 2013). We modify the model to simulate the effects of dedifferentiation and system segregation reduction through neural population reduction (NPR) and representation localization reduction (RLR). Our model provides new insight into the mechanisms underlying neural dedifferentiation and system segregation through the unique contributions of neural specificity on task performance.

Neural Dedifferentiation in Cognition

Cognitive function depends on the ability of neural systems to process and represent information in a specialized manner. Neural dedifferentiation refers to a reduction in the distinctiveness and selectivity of neural responses, wherein the difference in activation between preferred and non-preferred stimuli diminishes. This decline in neural specificity results from a loss of specialized neural characteristics that impair neural signalling, compromising the precision and fidelity of neural representations (Koen et al., 2019; Koen & Rugg, 2019; Voss et al., 2008). Koen et al. (2019) demonstrate this by computing a regional Differentiation Index (DI) for the parahippocampal place area (PPA) and lateral occipital cortex (LOC) regions of interest (ROIs).

DI is calculated using the means and variances of the across-trial BOLD responses of both the preferred and non-preferred image type in a categorical memory task. This is the difference between mean preferred (μ_{Pref}) and non-preferred ($\mu_{\text{Non-Pref}}$) image BOLD responses divided by their pooled standard error:

$$\text{DI} = \frac{\mu_{\text{Pref}} - \mu_{\text{Non-Pref}}}{\sqrt{\frac{\sigma^2_{\text{Pref}} + \sigma^2_{\text{Non-Pref}}}{2}}} \quad (1)$$

The greater the mean difference and the lower the error, the greater the DI. A high DI indicates lower neural dedifferentiation, where the difference between selection of preferred and non-preferred image types is highest.

Extensive research on neural dedifferentiation has focused on aging, where declines in neural specificity have been linked to impairments across multiple cognitive domains, including visuospatial ability, perception, memory, and cognitive control (de Frias, Lövdén, Lindenberger, & Nilsson, 2007). In younger individuals, neural responses are typically more specialized, with distinct brain regions selectively engaged in specific cognitive processes. However, in older adults, these previously distinct neural responses become more generalized, leading to overlapping activation patterns across different tasks and stimuli (Koen & Rugg, 2019).

This age-related decline in neural specificity can be attributed to two distinct mechanisms: broadening, where category-selective neurons increasingly respond to non-preferred stimuli; and attenuation, where activation to preferred stimuli diminishes. Park et al. (2012) investigate these processes using functional magnetic resonance imaging (fMRI) to examine neural dedifferentiation in the face-processing network, analyzing both the core face network, which includes the bilateral fusiform face area (FFA), occipital face area (OFA), and superior temporal sulcus (STS), and the extended face network, encompassing the amygdala (AMG), inferior frontal gyrus (IFG), and orbitofrontal cortex (OFC). Their findings demonstrate that dedifferentiation in the core face network primarily reflects broadening, as older adults exhibit increased activation to non-preferred stimuli, such as houses, indicating a decline in the precision of category-selective neurons. In contrast, the extended face network, particularly the prefrontal cortex and amygdala, shows attenuation, with reduced activation to preferred stimuli, such as faces, suggesting a diminished neural response with age. These findings highlight the complexity of neural dedifferentiation and its varied effects across different neural networks and regions of interest.

Cognitive aging not only introduces neural dedifferentiation, reducing the specificity of local neural representations, but also alters the organization of large-scale functional networks. As age-related decline progresses, previously distinct functional neural systems become less segregated, leading to increased cross-talk between cognitive networks and greater susceptibility to interference (Geerligs, Renken, Saliasi, Maurits, & Lorist, 2015; Rieck, Baracchini, Nichol, Abdi, & Grady, 2021). However, not all individuals experience equivalent cognitive decline, suggesting that some network-level properties may mitigate the impact of dedifferentiation (Rieck et al., 2021). One such property is system segregation, which maintains distinct functional boundaries within and between networks, preserving cognitive efficiency.

System Segregation and Cognitive Performance

System segregation refers to the degree to which subnetworks within a larger network maintain independence from one another (Wig, 2017). Chan et al. (2014) developed a method to estimate system segregation in the brain using Resting State Functional Connectivity (RSFC) data. An existing RSFC map is used to compose nodes using a winner-takes-all approach. These nodes are labelled with their associated functional system, and activation in nodal regions is used to measure within and between system correlations. Finally, these correlations are used in the System Segregation (SS) formula as an estimate of system segregation.

SS is the difference between mean within-system correlation ($\overline{Z}_{\mathrm{w}}$) and mean between-system correlation ($\overline{Z}_{\mathrm{b}}$), divided by the mean within-system correlation ($\overline{Z}_{\mathrm{w}}$):

$$\mathrm{SS} = \frac{\overline{Z}_{\mathrm{w}} - \overline{Z}_{\mathrm{b}}}{\overline{Z}_{\mathrm{w}}} \tag{2}$$

As age increases, within-system correlation reduces and between-system correlation rises. This estimate captures the degree to which distinct neural communities maintain independent processing, while still contributing to the broader network's functional integrity. Effective network function relies on striking a balance between maintaining subnetwork segregation, which preserves specialized processing, and allowing integration within and between subnetworks to facilitate communication and coordination. Key characteristics of segregated systems include dense intraconnections, where neural communities within the same subnetwork exhibit strong connectivity, and sparse interconnections, where communication between different subnetworks is comparatively weaker, but necessary for distributed processing.

This leads to a subtle discrepancy between Wig (2017)'s description of system segregation and the constituent variables used in Chan et al. (2014)'s estimate. Wig (2017)'s system segregation is described the same way neural differentiation is by Koen et al. (2019), where the focus is on differences in neural activation. Chan et al. (2014)'s estimate changes the focus to the difference between interregional and intraregional neural activation. DI and SS both rely on similar statistical methods to assess differences in fMRI BOLD responses, but SS is finer-grained, correlating multiple subnetwork means to capture these intraregional differences. Considering some mean neural connectivity can be expected in a particular region, a reduction to within-system correlation can be said to capture neural population reduction (NPR). Additionally, the overlap between the SS nominator with DI's nominator suggests both SS and DI capture some aspect of neural specificity.

Research indicates that young adults exhibit high system segregation, characterized by strong within-network connectivity and minimal interference between networks (Geerligs et al., 2015). In contrast, aging is associated with increased cross-network connectivity, particularly between the default mode network (DMN) and task-positive networks such as the

frontoparietal control network. This loss of network distinctiveness contributes to slower cognitive processing, reduced task-switching efficiency, and greater susceptibility to distraction (Chan et al., 2014; Geerligs et al., 2015).

As aging progresses, previously segregated networks become more interconnected, leading to functional dedifferentiation at the network level (Rieck et al., 2021). This weakening of functional boundaries results in diminished task-relevant activation, making it more difficult for older adults to suppress irrelevant information and maintain goal-directed behaviour (Geerligs et al., 2015).

Despite these age-related declines, some older adults maintain higher levels of system segregation, which has been linked to better cognitive function (Rieck et al., 2021). These findings suggest that preserving system segregation may serve as a protective factor against cognitive aging, supporting more effective neural processing and task performance.

Nengo and the Neural Engineering Framework

The Neural Engineering Framework (NEF) provides a systematic approach to constructing large-scale, biologically plausible neural models of cognition. It operates as a framework for neural simulation, allowing researchers to define neuron properties, representational values, and computational functions, while mathematically solving for synaptic connection weights that achieve these transformations (Eliasmith & Anderson, 2003). One of the core strengths of the NEF is its ability to model neural connections of different kinds (feedforward and recurrent) while enabling the construction of dynamical systems (integrators, oscillators and control mechanisms).

The NEF consists of three fundamental principles: representation, transformation, and dynamics (Eliasmith & Anderson, 2003; Voelker et al., 2021). The representation principle enables the encoding of continuous values through the activity of neural populations such that neural ensembles represent signals. The transformation principle allows for the computation of arbitrary functions using weighted synaptic connections, transforming representations through connections between neural ensembles. The dynamics principle supports the modelling of time-dependent processes through the integration of control theory, which is particularly beneficial for applications like working memory and motor control. These principles collectively allow for the construction of models that closely resemble real brain function, constrained by biological data.

Nengo is a Python-based computational modelling platform designed for constructing and simulating large-scale neural networks using the Neural Engineering Framework (NEF). It streamlines the process of defining, running, and analyzing complex neural simulations, making it an accessible and efficient tool for researchers. With its user-friendly interface, Nengo enables users to specify neural populations, define transformations, and simulate interactions with environmental stimuli, facilitating the study of cognitive processes and neural dynamics (Sharma, Aubin, & Eliasmith, 2016).

Nengo integrates biologically inspired learning rules, allowing it to simulate adaptive and plastic neural circuits that mimic the way real biological systems learn and change over time (Stewart, Bekolay, & Eliasmith, 2012). This capability has been instrumental in modeling reinforcement learning in the basal ganglia, where Nengo successfully replicates key neurophysiological properties observed in biological decision-making systems (Stewart, Bekolay, & Eliasmith, 2012).

The Semantic Pointer Architecture

The Semantic Pointer Architecture (SPA) was built to facilitate cognitive model construction using Nengo (Eliasmith, 2013). SPA uses a Vector Symbolic Architecture (VSA; Kleyko, Rachkovskij, Osipov, & Rahimi, 2022, 2023) to encode structured information in a high-dimensional latent space. In a VSA, information is encoded as vectors, and complex relationships are captured through mathematical operations that allow for the storage, retrieval, and manipulation of information in the latent space. In NengoSPA, this is accomplished using Holographic Reduced Representations (HRRs; Plate, 1995), which defines a binding operation over vectors that allows vector-symbols to be combined into vector-symbolic expressions.

SPA relies on the Semantic Pointer Hypothesis (SPH), which serves as a bridge between the low-level neural details of the NEF and higher-level cognition (Eliasmith, 2013). The SPH suggests that neural systems encode, process, and manipulate structured information using semantic pointers. Semantic pointers are high-dimensional vector representations that can be implemented as patterns of neural activity and represent arbitrarily complex symbolic information in a manner that is dynamically modifiable by the neural system.

The Nengo implementation of SPA is known as NengoSPA (Applied Brain Research, 2025). A major advantage of NengoSPA is its modular and extensible architecture. This modularity supports the construction of large-scale cognitive architectures, such as Spaun, one of Nengo's most notable applications (Stewart, Choo, & Eliasmith, 2012). Spaun is a large-scale brain model capable of performing high-level cognitive tasks, including working memory, problem-solving, action selection, and motor control. It processes visual stimuli, stores information in memory, and generates motor output by controlling a simulated robotic arm, demonstrating how large-scale neural systems coordinate multiple cognitive functions without requiring external reprogramming (Eliasmith et al., 2012).

Models of Neural Impairments

Understanding how the brain encodes, processes, and retrieves information is a fundamental challenge in computational neuroscience. The NEF and SPH provide a biologically plausible approach to modelling cognitive functions, enabling simulations of perception, memory, decision-making, motor

control, and cognitive decline. Computational models built in Nengo have been used to investigate various neurological impairments, including Parkinson's disease, Alzheimer's disease, and age-related cognitive slowing. By leveraging spiking neural networks (SNNs) and high-dimensional vector representations, these models offer insights into how neural circuits break down in neurodegenerative disorders.

Senft, Stewart, Bekolay, Eliasmith, and Kröger (2018) developed a spiking neural network model of the basal ganglia-thalamus-cortex loop to study speech sequencing deficits in Parkinson's disease. Their findings showed that dopaminergic depletion leads to syllable sequencing errors, and deep brain stimulation (DBS) reduces these impairments by inhibiting the subthalamic nucleus (STN) and globus pallidus internus (GPi). Wijs (2022) examined pattern separation in the hippocampus using a spiking model of the EC-DG-CA3 circuit, showing that Alzheimer's disease-related synaptic changes increase pattern overlap, leading to memory encoding deficits. Ahmed, Lytton, Stewart, and Crystal (2024) explored age-related cognitive slowing with Nengo-based models of the Stroop task, demonstrating that axonal loss and input noise slow processing, while increased feedback preserves memory at the cost of further slowing.

The integration of semantic pointers, vector symbolic architectures (VSAs), and biologically inspired learning mechanisms provides a unified computational framework for studying cognitive decline. We used these same tools in our own NengoSPA simulations to capture features of neural specificity.

Models & Simulations

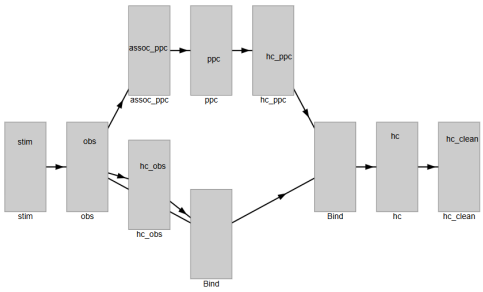
To demonstrate neural dedifferentiation and system segregation in NengoSPA, we modelled two interventions. The first intervention involved a reduction in neural specificity directly through representation localization reduction (RLR), accomplished by the de-localization of vocabularies between states in computationally modelled brain regions. With less localized representations across multiple neurons, performance errors should arise from an inability to differentiate between the target representation and other, similar ones. This is similar to an individual with reduced system segregation. The second intervention captured within-system correlation reduction in our simulations using neural population reduction (NPR). A reduction in neural population lowers the degrees of freedom with which to encode a representation, due to less neural connections on account of the neural loss. Since these models are functional, NPR is a stand-in for any neural dysfunction that interrupts the neuron's activity.

To assess the effects of these two interventions, we constructed a restricted Go/No-go cognitive task. In this task, a set of stimuli are provided with action rules, wherein one stimulus is the GO condition and another is the NO-GO condition. Normally, when the GO condition is presented, some response is required by the participant. When the NO-GO condition is presented, response inhibition is required. In our models, processing of the stimulus is cut off before action selection or any prefrontal monitoring or control.

In these models, stimuli are passed to neural ensembles as semantic pointers, allowing for encoding, storage, decoding, and manipulation of the representations to facilitate cognitive processing; the output of which is some decision, prior to an action selection loop. In each model, rule matching between internal representations and perception is assumed to happen prior to action selection, but the cortico-basal ganglia-thalamo-cortical (CBGTC) loop was not modelled. This assumption aligns with the theoretical considerations taken in the construction of both Spaun (Stewart, Choo, & Eliasmith, 2012) and Borst, Aubin, and Stewart (2023)'s whole-task brain model. This means our model only requires a decision to be made, without a behavioural response. No inhibition of an action is necessary, and no prefrontal monitoring or control is present.

Input to the hippocampus (hc) state will be compared to its output, providing a similarity value for the model's decision. Each model will run for 6 seconds over 50 simulations with 8 similarity samples taken per simulation, first at 0.375s and then at each additional interval of 0.75s in accordance with the presentation of our two stimuli: VOW or CONS. VOW, standing for vowel, indicates the GO decision should be made, while CONS, standing for consonant, indicates the NOGO decision should be made.

Figure 1: Go/No-go Model in NengoSPA



In Figure 1, the stimuli serve as an input that is passed to an observation state (obs). The obs state encodes the stimuli as one of two semantic pointers, VOW for vowels, and CONS for consonants. Two equivalence rules have *a priori* representation as semantic pointers embedded in the posterior parietal cortex (ppc) state. These rules are "VOW = GO" and "CONS = NOGO". Connective states (hc_obs & hc_ppc) are used with unique semantic pointer vocabularies to simulate interregional and intraregional neural specificity between the obs and ppc states with the hippocampal (hc) state. The hc state receives a representation consisting of a binding of both the hc_ppc state and the negative of the hc_obs state. VOW and GO share a vector due to the equivalence rule, so the negative of the hc_obs state removes VOW during binding. GO is passed to a clean-up memory.

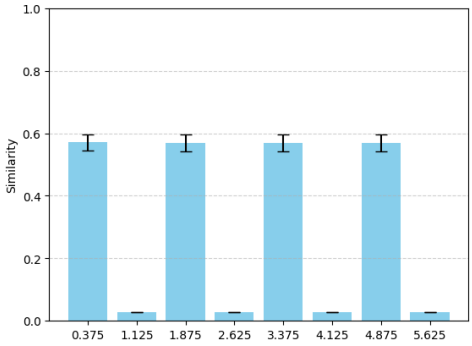
A clean-up memory is a system that corrects any repre-

sentation to one of a set of canonical representations in the clean-up memory's vocabulary. The clean-up memory in our model has no threshold, which means that the model always outputs one of the symbols in the vocabulary; however, were a threshold added, low semantic pointer similarity could fail to pass threshold and fail to activate neural populations in the clean-up memory.

Results

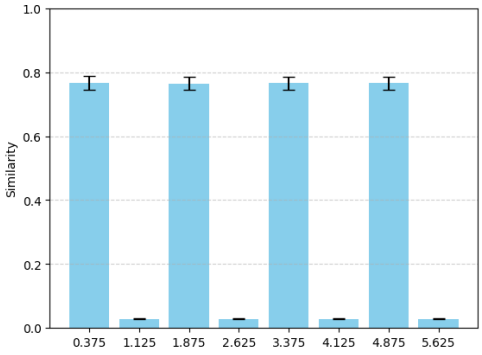
In Figure 2, we see a medium similarity to GO at each sampled timestamp, averaged over 50 simulations (M = 0.57, SE $<$ 0.01). Similar results were found for NOGO (M = 0.54, SE $<$ 0.01).

Figure 2: Control Model Mean GO Similarity Over Time



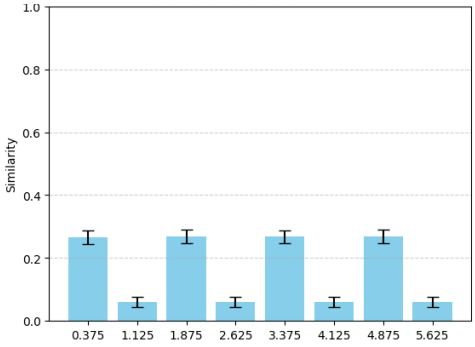
In Figure 3, we see a high similarity to GO at each sampled timestamp, averaged over 50 simulations (M = 0.77, SE $<$ 0.01). Similar results were found for NOGO (M = 0.73, SE $<$ 0.01). Additionally, VOW demonstrated a high similarity (M = 0.77, SE $<$ 0.01) during each GO stimulus, and CONS demonstrated a high similarity at each NOGO stimulus (M = 0.73, SE $<$ 0.01). This suggests co-activation of GO and VOW responses to the GO stimulus, and co-activation of NOGO and CONS to the NOGO stimulus.

Figure 3: RLR Model Mean GO Similarity Over Time



In Figure 4, we see a low similarity to GO at each sampled timestamp, averaged over 50 simulations (M = 0.27, SE $<$ 0.01). Similar results were found for NOGO (M = 0.27, SE $<$ 0.01).

Figure 4: NPR Model Mean GO Similarity Over Time



Each simulation is independent of the previous trials. Tests for normality and homoscedasticity were performed. A Shapiro-Wilk test was performed, demonstrating the data violated the assumption of normality (Table 1).

Table 1: Shapiro-Wilk Test for each Model & Stimulus

	Shapiro-Wilk	p
GO Control	0.61	$<$0.01
GO RLR	0.68	$<$0.01
GO NPR	0.73	$<$0.01
NOGO Control	0.65	$<$0.01
NOGO RLR	0.72	$<$0.01
NOGO NPR	0.74	$<$0.01

A Levene's test demonstrated homoscedasticity was also violated, except for NOGO-NOGO (Table 2).

Table 2: Levene's Test for each Model & Stimulus

	VOW	CONS	GO	NOGO
GO	323.73	46.80	6.64	3.833
p	$<$0.01	$<$0.01	$<$0.01	0.02
NOGO	25.06	251.79	352.90	0.019
p	$<$0.01	$<$0.01	$<$0.01	0.98

Due to these violations, a Welch's ANOVA was performed demonstrating high significance with a very large effect size suggesting group membership at each timestamp describes 73% of the variance in similarity scores, $F(2,352) = 427.01$, $p < 0.01$, $\eta^2 = 0.73$.

A Games-Howell post-hoc test confirms this (Table 3), demonstrating highly significant differences in variance between all model similarity scores. For both Control-NPR and NPR-RLR comparisons, effect sizes are very large, while the Control-RLR comparison had a medium effect size.

Table 3: Games-Howell Test

Comparison	Abs. Mean Diff.	p	Hedges' g
Control-NPR	0.03	<0.01	-2.87
Control-RLR	<0.01	<0.01	-0.66
NPR-RLR	0.03	<0.01	2.80

Discussion

Results from our model comparison suggests that the biological properties RLR and NPR represented could be candidate mechanisms for neural specificity. RLR directly modeled a loss in system segregation, directly modelling the effects of neural broadening. While this was a top-down approach for the model, the biological equivalent is a bottom-up process whereby interregional activity is lost, in favour of interregional activity. In our model, NPR is an actual reduction in the number of neurons; however, our model is functional and any loss of a neuron's ability to send and receive signals would be equivalent. The RLR model demonstrated much higher similarity than the control; however, a co-activation was found with each of the paired stimuli (VOW with GO; CONS with NOGO). This comports with the commission errors found with neural broadening. NPR demonstrated a similar pattern, where similarity was significantly lower than with the other models. Were the clean-up memory's threshold set to 0.5, the similarity would've been too low to provide a response.

The impact of NPR on representation and performance explains within-system correlation well, and the cause of NPR's loss in neural function can come from multiple sources such as illness, senescence, or brain trauma. Nonetheless, a reduction in neural specificity cannot result from random impairments to neural function alone, because those impairments fail to account for all characteristics of performance decline. The very fact commission errors and neural broadening happens suggests neural specificity reduction results from another, different mechanism, which is precisely why we modelled RLR. It is likely neural specificity reduction results from a kind of passive, functional compensation to overcome the loss in neural function. As neural function decreases, processes like dendritic arborization do not cease. The continued growth of new connections may not be enough to compensate for the lost functional activity, especially when cumulative damage to representations builds as functional impairment continues. Instead, it seems likely neural specificity loss is partially explained by compensatory neuroplasticity that can't quite fill the representational gaps completely.

Future research into these mechanisms and their relevance in explicating cognitive decline should focus on the effects of neural dysfunction (NPR) on modularity and neural redundancy. Modularity is calculated by comparing the observed fraction of within-community connections to the expected fraction if connectivity were randomly distributed across the network (Newman, 2006). Comparisons between randomly (or uniformly) distributed and regionally targeted NPR could indicate idiosyncratic cognitive impairments. Future research on RLR should focus on whole-task brain models to see if co-activation of adversarial representations causes unique impairments in different neural systems. Additional work could be done investigating intraregional effects of RLR, to investigate finer-grained neural specificity, similar to what is done with neuroimaging techniques like voxel-wise modelling (Huth, De Heer, Griffiths, Theunissen, & Gallant, 2016). An extension could be done to test whether DI and SS formulae can be applied to NengoSPA models. Lastly, it may be worth exploring patients with traumatic brain injury or stroke to see if there is a shift from omission errors to commission errors during or following recovery. If so, this would suggest neural specificity reduction results from both a loss of neural function and the inability of neuroplasticity to completely compensate, leading to less distinct representations and partially impaired information processing in the brain.

References

Ahmed, S., Lytton, W. W., Stewart, T. C., & Crystal, H. (2024). Computational models of age-associated cognitive slowing. *bioRxiv*, 2024–06.

Applied Brain Research. (2025). *Nengospa.* Retrieved from `https://www.nengo.ai/nengo-spa/`

Borst, J. P., Aubin, S., & Stewart, T. C. (2023). A whole-task brain model of associative recognition that accounts for human behavior and neuroimaging data. *PLOS Computational Biology*, *19*(9), e1011427.

Chan, M. Y., Park, D. C., Savalia, N. K., Petersen, S. E., & Wig, G. S. (2014). Decreased segregation of brain systems across the healthy adult lifespan. *Proceedings of the National Academy of Sciences*, *111*(46), E4997–E5006.

de Frias, C. M., Lövdén, M., Lindenberger, U., & Nilsson, L.-G. (2007). Revisiting the dedifferentiation hypothesis with longitudinal multi-cohort data. *Intelligence*, *35*(4), 381–392.

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition.* OUP USA.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *science*, *338*(6111), 1202–1205.

Geerligs, L., Renken, R. J., Saliasi, E., Maurits, N. M., & Lorist, M. M. (2015). A brain-wide study of age-related changes in functional connectivity. *Cerebral cortex*, *25*(7), 1987–1999.

Huth, A. G., De Heer, W. A., Griffiths, T. L., Theunissen, F. E., & Gallant, J. L. (2016). Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, *532*(7600), 453–458.

Kleemeyer, M. M., Polk, T. A., Schaefer, S., Bodammer, N. C., Brechtel, L., & Lindenberger, U. (2017). Exercise-induced fitness changes correlate with changes in neural specificity in older adults. *Frontiers in Human Neuroscience*, *11*, 123.

Kleyko, D., Rachkovskij, D., Osipov, E., & Rahimi, A. (2023). A survey on hyperdimensional computing aka vector symbolic architectures, part ii: Applications, cognitive models, and challenges. *ACM Computing Surveys*, *55*(9), 1–52. doi: 10.1145/3558000

Kleyko, D., Rachkovskij, D. A., Osipov, E., & Rahimi, A. (2022). A survey on hyperdimensional computing aka vector symbolic architectures, part i: Models and data transformations. *ACM Computing Surveys*, *55*(6), 1–40. doi: 10.1145/3538531

Koen, J. D., Hauck, N., & Rugg, M. D. (2019). The relationship between age, neural differentiation, and memory performance. *Journal of Neuroscience*, *39*(1), 149–162.

Koen, J. D., & Rugg, M. D. (2019). Neural dedifferentiation in the aging brain. *Trends in cognitive sciences*, *23*(7), 547–559.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the national academy of sciences*, *103*(23), 8577–8582.

Park, J., Carp, J., Hebrank, A., Park, D. C., & Polk, T. A. (2010). Neural specificity predicts fluid processing ability in older adults. *Journal of Neuroscience*, *30*(27), 9253–9259.

Park, J., Carp, J., Kennedy, K. M., Rodrigue, K. M., Bischof, G. N., Huang, C.-M., ... Park, D. C. (2012). Neural broadening or neural attenuation? investigating age-related dedifferentiation in the face network in a large lifespan sample. *Journal of Neuroscience*, *32*(6), 2154–2158.

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

Rieck, J. R., Baracchini, G., Nichol, D., Abdi, H., & Grady, C. L. (2021). Reconfiguration and dedifferentiation of functional networks during cognitive control across the adult lifespan. *Neurobiology of Aging*, *106*, 80–94.

Senft, V., Stewart, T. C., Bekolay, T., Eliasmith, C., & Kröger, B. J. (2018). Inhibiting basal ganglia regions reduces syllable sequencing errors in parkinson's disease: a computer simulation study. *Frontiers in computational neuroscience*, *12*, 41.

Sharma, S., Aubin, S., & Eliasmith, C. (2016). Large-scale cognitive model design using the nengo neural simulator. *Biologically Inspired Cognitive Architectures*, *17*, 86–100.

Stewart, T. C., Bekolay, T., & Eliasmith, C. (2012). Learning to select actions with spiking neurons in the basal ganglia. *Frontiers in neuroscience*, *6*, 2.

Stewart, T. C., Choo, F.-X., & Eliasmith, C. (2012). Spaun: A perception-cognition-action model using spiking neurons. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 34).

Voelker, A. R., Blouw, P., Choo, X., Dumont, N. S.-Y., Stewart, T. C., & Eliasmith, C. (2021). Simulating and predicting dynamical systems with spatial semantic pointers. *Neural Computation*, *33*(8), 2033–2067.

Voss, M. W., Erickson, K. I., Chaddock, L., Prakash, R. S., Colcombe, S. J., Morris, K. S., ... Kramer, A. F. (2008). Dedifferentiation in the visual cortex: an fmri investigation of individual differences in older adults. *Brain research*, *1244*, 121–131.

Wig, G. S. (2017). Segregated systems of human brain networks. *Trends in cognitive sciences*, *21*(12), 981–996.

Wijs, C. (2022). *The influence of connectivity sparseness and alzheimer's disease on pattern separation in a spiking neuron model of the ec-dg-ca3 hippocampal circuit*. Unpublished doctoral dissertation, University of Groningen.

Unpacking the Election: Unpacking Effects in Electoral Prediction Markets

Christopher R. Fisher (christopher.fisher.27.ctr@us.af.mil)
Parallax Advanced Research
Beavercreek, OH 45324 USA
Kevin Schmidt (kevin.schmidt.15@afrl.af.mil) Taylor Curley(taylor.curley@us.af.mil)
Air Force Research Laboratory
Wright Patterson AFB, OH 45433 USA

Abstract

Prediction markets have been used to forecast a variety of future events, ranging from sports to geopolitical events. The market price is often interpreted as a crowd-sourced probability estimate of a future event. While much of the literature has focused on understanding factors that affect accuracy, much less research has examined the degree to which prediction markets adhere to the rules of probability theory. Existing research on the unpacking effect, which occurs when probability judgments of disjoint sub-events do not sum to the probability of the general event, has produced mixed results. In a popular online platform, we found evidence of unpacking effects in prediction markets for the 2024 US elections. As a proof of concept, we demonstrate that a multi-agent model based on principles of quantum cognition can produce several qualitative patterns of unpacking effects. We conclude with a discussion of limitations of the model and directions for future research.

Keywords: wisdom-of-crowds; prediction markets; forecasting; quantum cognition; decision-making; unpacking effects

Introduction

An extensive body of research indicates that aggregating judgments across a large number of individuals leads to more accurate estimates, a phenomenon known as *the wisdom of crowds* (Surowiecki, 2004). In many cases, an aggregated judgment is more accurate than the best individual judge. Judgments can be aggregated through a variety of methods, including averaging independent judgments across individuals (Surowiecki, 2004), team interactions (Mellers et al., 2014), prediction markets (Berg et al., 2008), or a combination thereof (Dana et al., 2019).

Our focus will be on prediction markets in which individuals buy and sell shares representing the outcome of a future event, such as the winner of a sports competition (Page & Clemen, 2013), the winner of an election (Berg et al., 2008), or the replication of an experiment (e.g., Dreber et al., 2015). As an example, consider a person who bought a share for candidate A winning an election for .70 dollars. If candidate A wins, the shareholder receives 1 dollar. However, if candidate A does *not* win, the shareholder receives nothing, and losses the initial investment. In a prediction market, the wisdom of crowds emerges through transactions between individuals with heterogeneous knowledge who are incentivized to maximize profit, ultimately converging on a market price that reflects the pooled knowledge of all participants.

Much of the research pertaining to prediction markets has focused on assessing accuracy or devising methods to improve accuracy. For example, prediction markets were more accurate for long time horizons than polling in predicting the outcome of US elections between 1988 and 2004 (Berg et al., 2008). Across a wide range of prediction markets involving political and sporting events, Page & Clemen (2013) observed the favorite-longshot bias, whereby high probability events tend to be under-priced and low probability events tend to be over-priced. However, as the time horizon decreased, the prices became better calibrated. Similarly, for prediction markets involving European elections and referendums, Arnesen & Strijbis (2015) found evidence of the favorite-longshot bias. Additionally, their analysis attributed the favorite-longshot bias, in part, to the use logarithmic scoring rule in the market maker (an algorithm used to promote liquidity in markets), which causes prices to change more slowly as they approach the end points of 0 or 1. Dana et al. (2019) compared prediction markets to individual estimates which were aggregated using an algorithm that adjusted for regression towards the mean and weighted individuals according to prior accuracy. The aggregation algorithm was at least as accurate as the prediction market, and combining the methods resulted in additional improvement, indicating some degree of inefficiency in the prediction market. Taken together, the literature indicates that prediction markets are useful, albeit imperfect, forecasting tools.

Based on the empirical findings above and theoretical analysis (e.g., Wolfers & Zitzewitz, 2006), the market price is commonly interpreted as a crowd-sourced probability estimate for the occurrence of an event. One implication of interpreting market prices as probability estimates is that market prices should conform to the rules of probability theory. Although variety of violations of probability theory have been observed in the estimates of individuals (e.g, Tversky & Kahneman, 1983; Tversky & Koehler, 1994; Busemeyer et al., 2011), much less is known about the coherence and logical consistency of prediction markets. The limited research investigating coherence of prediction markets has yielded mixed results. Lee et al. (2009) investigated the conjunction fallacy and unpacking effect in a wide variety of prediction markets. A conjunction fallacy occurs when the judged probability of joint events exceeds the probability of one or both of its constituent events (Tversky & Kahneman, 1983), e.g. $p(a \wedge b) > p(a)$. An unpacking effect occurs when an event is unpacked or decomposed into disjoint sub-events which do not sum to the original event (Tversky & Koehler, 1994), e.g., $p(a) \neq p(a_1) + p(a_2) + \cdots + p(a_n)$. Generally speaking, the market prices tended to adhere to the rules of probability theory, only producing an unpacking effect in a few transient and

isolated instances (Lee et al., 2009). By contrast, Sonnemann et al. (2013) found evidence of unpacking effects in prediction markets for sporting competitions in both controlled laboratory experiments and naturalistic field data.

As prediction markets have become increasingly popular and accessible, one question that naturally arises is whether unpacking effects might be observed in popular prediction markets with diverse participants. A related question is through which cognitive mechanisms could unpacking effects potentially emerge? To the best of our knowledge, very few efforts have been made to analyze prediction markets through the lens of a cognitive model. To fill these gaps in the literature, we investigated unpacking effects in prediction markets for the 2024 US elections and developed a proof-of-concept model based on principles from quantum cognition (e.g., Busemeyer et al., 2011). We selected quantum cognition as our modeling framework because it can produce several types of unpacking effects (Franco, 2010), and other violations of classical probability theory (Busemeyer et al., 2011).

Overview

Our primary goal in this paper is to investigate unpacking effects in prediction markets and to provide a viable explanation for future research to explore. In what follows, we begin with a review unpacking effects in the judgments of individuals, including subadditivity and superadditivity. Next, we describe the mechanics of Polymarket, an online prediction market platform. We then describe how several markets pertaining to the US 2024 elections can be leveraged as tests of unpacking effects. Our analysis of these markets reveals evidence of multiple sustained unpacking effects. As a potential explanation for these findings, we introduce an agent based model of the prediction market in which agents evaluate event probabilities according principles of quantum cognition (e.g., Busemeyer et al., 2011). We conclude with a discussion of key findings, limitations, and future directions.

Unpacking Effects

An unpacking effect occurs when the sum of probability judgments across disjoint partitions of an event is not equal to the judged probability of the event (Tversky & Koehler, 1994; Sloman et al., 2004; Fiedler et al., 2009). As an example, previous research found that the probability judgments for death by natural cause were lower on average than the sum of separate probability judgments for specific types of natural causes of death: death by cancer, death by heart disease, and death by all other natural causes (Tversky & Koehler, 1994). Unpacking effects constitute a violation of a property of probability theory called additivity. Suppose an event e is unpacked or partitioned into a set of mutually exclusive and exhaustive sub-events: $E = \{e_1, e_2, \ldots, e_n\}$. Formally, additivity is defined as:

$$p(e) = \sum_{e_i \in E} p(e_i). \tag{1}$$

Unpacking effects lead to one of two violations of additivity. The first type, which is the most commonly reported, is subadditivity (Tversky & Koehler, 1994), defined as: $j(e) \leq \sum_{e_i \in E} j(e_i)$, where the judgment function $j(\cdot)$ does not require judgments to adhere to the rules of probability theory. The second type is superadditivity, defined as $j(e) \geq \sum_{e_i \in E} j(e_i)$. Some evidence suggests superadditivity is likely to occur when the partition includes an atypical sub-event (Sloman et al., 2004).

Polymarket

Polymarket is a crypto-based online prediction market platform in which users can buy and sell shares for a wide range of future events, such as the winner of a football match, the Federal Reserve's decision to change interest rates, and the winner of an election. In a given market, users buy and sell shares for binary events. A *yes* share corresponds to occurrence of event e, whereas a *no* share corresponds to the occurrence of the complementary event $\bar{e}$ (Polymarket, 2024). At time t, the market price of a share for event $x \in \{e, \bar{e}\}$ is $p_x(t) \in [0,1]$, which pays 1 if event x occurs and 0 otherwise. A person's subjective probability of event x is denoted as $j(x)$. Letting V_x be a random variable representing the eventual payoff of a share for event x, the subjective expected value of a share purchased at price s_x is $\mathbb{E}[V_x] = j(x) \cdot [1 - s_x] - [1 - j(x)] \cdot s_x = j(x) - s_x$. From a normative perspective, a person should purchase a share if $p_x(t) < j(x)$.

Polymarket is classified as continuous double auction whereby bids and asks are tabulated in an order book and an exchange occurs when the bid and ask price match (see details below; Polymarket, 2024). Users can visualize the order book with a depth chart showing the volume of bid and ask prices in descending order. The difference between the minimum ask and maximum bid is known as the spread. A transaction occurs when the spread is zero. Users can submit orders (bids or asks) using one of two primary methods: a market or a limit (Polymarket, 2024). With the market method, users buy or sell shares immediately at the current market price. By contrast, the limit method allows users to submit orders which are executed at a later date if the market price converges to the bid or ask price. Polymarket also provides incentives to promote liquidity in markets (Polymarket, 2023).

In what follows, we describe the conditions under which a transaction occurs. To facilitate the description, we will introduce some notation and definitions. Let b_e and a_e be a bid price and ask price for event e, respectively. For binary events, a bid (ask) is equivalent to 1 minus an ask (bid) for the complementary event, e.g. $b_e = 1 - a_{\bar{e}}$. Given these definitions, a transaction occurs under three conditions: (1) if $b_{i,e} = a_{k,e}$, user i pays user $k \neq i$ the amount of $b_{i,e}$ in exchange for one share; (2) if $b_{i,e} + b_{k,\bar{e}} = 1$, a share for e is created and given to user i for the amount of $b_{i,e}$, and likewise a share for $\bar{e}$ is created and given to user k for the amount of $b_{k,\bar{e}}$; and (3) if $a_{i,e} + a_{k,\bar{e}} = 1$, user i receives $a_{i,e}$ and relinquishes 1 share

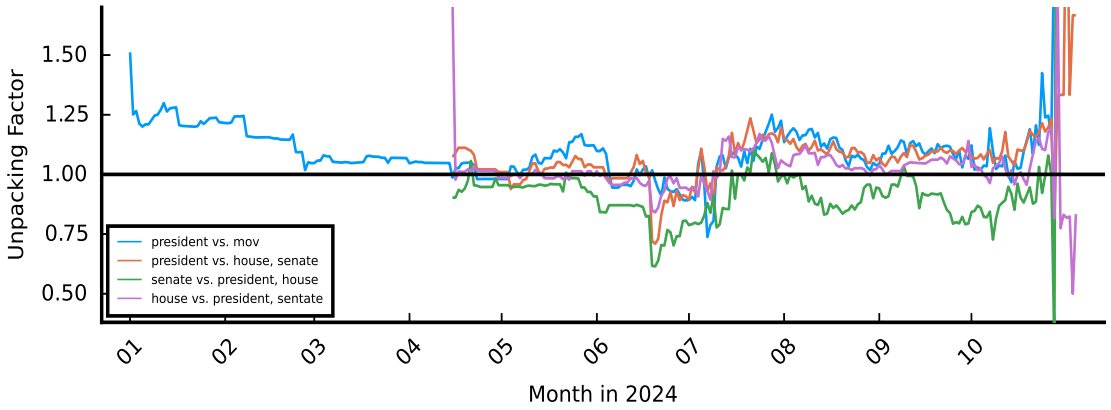


Figure 1: The unpacking factor for multiple market sets. Values greater than 1 correspond to subadditivity whereas values less than 1 correspond to superadditivity. mov: margin of victory. See text for details.

for event e, and user k receives $a_{k,\bar{e}}$ and relinquishes 1 share for event $\bar{e}$ (Polymarket, 2023).

A prediction market works in a similar fashion to a share market: as individuals with heterogeneous knowledge buy and sell shares, the market converges on an equilibrium or market price. With some caveats, the market price can be interpreted as crowd-sourced probability estimate for the occurrence of event (Wolfers & Zitzewitz, 2004). An important implication is that market prices should be constrained by rules of probability theory, including additivity as described above in Equation 1. In what follows, we will present evidence showing sustained periods of unpacking effects in prediction markets.

2024 Election Markets

One desirable feature of Polymarket is that it provides numerous opportunities for testing additivity of market-based probability estimates. Tests of additivity involve the comparison of two market sets: (1) a market set containing a single market for the binary event $\{(e,\bar{e})\}$, and (2) a market set in which event e is partitioned into n mutually exclusive and exhaustive sub-events: $\{(e_1,\bar{e}_1),(e_2,\bar{e}_2)\ldots(e_n,\bar{e}_n)\}$. The market for e vs. $\bar{e}$ was displayed by itself on a webpage, whereas markets for each binary sub-event e_i vs. $\bar{e}_i$ were grouped together on the same webpage and embedded within a full partition of the sample space, $E \subset \Omega$. Users were free to buy or sell shares in all of the markets.

For our analysis, we compared several market sets pertaining to the 2024 US elections for the presidency, senate, and house of representatives. Our description of the market sets follow the same order listed in the legend of Figure 1. In the first comparison, we compared a market for a Democrat winning the presidency to a set of markets in which a Democrat won the presidency with different margins of victory. For the remaining three, we compared a market for Democrats winning a given branch of government (presidency, senate, house) to a market set in which that event was unpacked into four sub-events in which Democrats or Republicans win the other two branches of government. For each of these comparisons, probability theory predicts the additive property described in Equation 1 will hold.

In the Figure 1, additivity is measured with a ratio called the unpacking factor (Tversky & Koehler, 1994), which is defined as $\text{uf} = \sum_{e_i \in E} j(e_i) \big/ j(e)$. Thus, additivity holds when $\text{uf} = 1$, subadditivity occurs when $\text{uf} > 1$, and superadditivity occurs when $\text{uf} < 1$. Examination of Figure 1 reveals multiple sustained violations of additivity. For example, the market set for Democrats and Republicans winning the electoral college with different margins of victory (purple) was consistently subadditive. However, subadditivity was less consistent and more attenuated for other markets. Markets tended to display subadditivity, with exception of the market for the senate (green), which tended to be superadditive. Overall, the prediction markets displayed evidence of sustained unpacking effects across time, but the effects were smaller in magnitude compared to what is reported for individual judgments (e.g., Tversky & Koehler, 1994).

Agent Based Model

We developed two agent based models using quantum cognition to contrast their differing predictions for unpacking effects in prediction markets. One model assumed beliefs are compatible, whereas the other assumed beliefs were incompatible. Both models consisted of 1,000 agents, each with an initial endowment of 500 dollars, and two market sets designed to examine unpacking effects. On each day, each agent submitted one order (ask or bid) in each available market. Agents submitted orders sequentially in a different random order on each day. We set the duration to the maximum duration in Figure 1, which was 305 days.

In both models, agents represented and evaluated their beliefs based on principles from quantum probability theory (QPT; Busemeyer et al., 2011). Quantum probability theory can be viewed as a generalization of classical probability theory in which certain properties such as commutativity do not necessarily hold (Busemeyer et al., 2011). In QPT,

beliefs are represented geometrically as sub-spaces within a vector space. Additionally, subjective probabilities are found by projecting a superposition state (i.e., a linear combination of basis vectors) onto the subspace spanned by the beliefs in question. An important distinction in QPT is whether beliefs are compatible. If beliefs are compatible, they can be represented with a single basis, in which case QPT reduces to classical probability theory and does not predict unpacking effects. However, if beliefs are incompatible, they are represented by different bases in a lower dimensional space, and are evaluated sequentially (Busemeyer et al., 2011). In this case, quantum cognition can account for unpacking effects and other violations of classical probability theory.

In the compatible quantum model (CQM), beliefs are jointly represented with a common basis. As such, the CQM satisfies the rules of classical probability theory, including additivity. By contrast, beliefs in the incompatible quantum model (IQM) cannot be represented jointly with a common basis, leading to violations of additivity. Intuitively, the different bases allow agents to view the markets from different perspectives based on available information. Our proposal is that, in principle, a qualitative change in information could cause agents to view the markets from a different perspective (i.e., basis), leading to periods of subadditivity or superadditivity.

Prediction Markets

Agents traded shares in two market sets designed to explore predictions for subadditivity and superadditivity. Although the nature of the events do not necessarily matter, event A could refer to a candidate winning the electoral college, and event B could refer to the same candidate achieving a plurality. The first market set consisted of one market for event A: $\{(A,\bar{A})\}$. The second market set consisted of four markets formed from the joint distribution over events A and B: $\{(A\wedge B,\overline{A\wedge B}),(A\wedge\bar{B},\overline{A\wedge\bar{B}}),(\bar{A}\wedge B,\overline{\bar{A}\wedge B}),(\bar{A}\wedge\bar{B},\overline{\bar{A}\wedge\bar{B}})\}$. According to probability theory, the market price for event A in the first market set should equal the sum of market prices for the unpacked events in the second market set: $p(A)=p(A\wedge B)+p(A\wedge\bar{B})$, which implies an unpacking factor of 1.

Belief Representation and Evaluation

The critical distinction between the CQM and the IQM is whether beliefs are compatible, which has implications for unpacking effects. As the name implies, beliefs in the CQM are compatible, meaning it is possible to represent the full 2×2 joint distribution of beliefs with a common basis. The CQM represents beliefs in 4D space using the following orthonormal basis: $\{|\mathrm{AB}\rangle,|\mathrm{A\bar{B}}\rangle,|\mathrm{\bar{A}B}\rangle,|\mathrm{\bar{A}\bar{B}}\rangle\}$. Because the joint probability distribution can be represented with a single basis (i.e. events are compatible), the probability judgments conform to classical probability theory, including additivity. An agent's initial state is represented as a superposition or linear combination of basis vectors: $|\psi\rangle=\psi_{\mathrm{AB}}|\mathrm{AB}\rangle+\psi_{\mathrm{A\bar{B}}}|\mathrm{A\bar{B}}\rangle+\psi_{\mathrm{\bar{A}B}}|\mathrm{\bar{A}B}\rangle+\psi_{\mathrm{\bar{A}\bar{B}}}|\mathrm{\bar{A}\bar{B}}\rangle$, where coefficients in ψ represent amplitudes. An agent's probability judgment is found by projecting the agent's state vector onto the subspace representing the target event and squaring the magnitude. For example, the probability judgment for $A\wedge B$ is given by $j(A\wedge B)=\|\mathbf{P}_{\mathrm{AB}}|\psi\rangle\|^2=\psi^2_{\mathrm{AB}}$, where the projection matrix for the event is computed as the outer product of the corresponding basis vector: $\mathbf{P}_{\mathrm{AB}}=|\mathrm{AB}\rangle\langle\mathrm{AB}|$. For each agent k, the square of the amplitudes (i.e., probabilities) are defined as: $\psi'_k\sim\mathrm{Dirichlet}([0.20,0.25,0.10,0.45]\cdot 20)$, such that $\psi_k=\psi_k'^{\odot\frac{1}{2}}$ and $\odot$ denotes an element-wise square root.

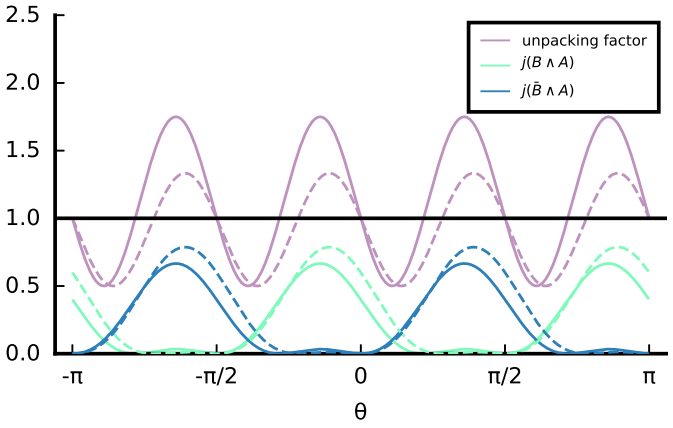


Figure 2: Predictions for the IQM as a function of rotation parameter θ. Solid lines correspond to $\rho=.40$ and dashed lines correspond to $\rho=.60$.

By contrast, in the IQM, the joint distribution of beliefs cannot be represented with a common basis. Instead, beliefs are represented in a reduced 2D space with two orthonormal bases: $\{|A\rangle,|A\rangle^{\perp}\}$ and $\{|B\rangle,|B\rangle^{\perp}\}$. One basis is obtained by rotating the other basis e.g., $|B\rangle=U(\theta)|A\rangle$, where U a unitary transformation and θ is the rotation angle in radians. Because the events are incompatible, they must be evaluated through a sequence of order-dependent projections. The probabilities of the two unpacked events are $j(B\wedge A)=\|\mathbf{P}_A\mathbf{P}_B|\psi\rangle\|^2$ and $j(\bar{B}\wedge A)=\|\mathbf{P}_A\mathbf{P}_{\bar{B}}|\psi\rangle\|^2$. By contrast, the probability of the packed event is defined as $j(A)=\|\mathbf{P}_A|\psi\rangle\|^2$. Importantly, expanding the packed term reveals its relationship to the unpacked terms: $j(A)=\|\mathbf{P}_A\mathbf{P}_B|\psi\rangle\|^2+\|\mathbf{P}_A\mathbf{P}_{\bar{B}}|\psi\rangle\|^2+\Delta$. The key difference is that the unpacked equation includes an interference term, Δ. Subadditivity occurs when $\Delta<0$ and superadditivity occurs when $\Delta>0$. For agent k, the initial state is defined as $|\psi_k\rangle=\left[\sqrt{\rho_k},-\sqrt{1-\rho_k}\right]$, with $\rho_k\sim\mathrm{Beta}(.45\cdot 20,(1-.45)\cdot 20)$. During the first 100 days, agents viewed the markets from the perceptive represented by the angle parameter $\theta_{k,1}\sim\mathrm{normal}(.90,.01)$. After 100 days, new information was disseminated to the agents, leading to a change in perspective represented by angle parameter $\theta_{k,2}\sim\mathrm{normal}(.40,.01)$. The distributions for θ_1 and θ_2 were selected to produce subadditivity and superadditivity, respectively. Our rationale for selecting day 100 to induce a change in perspective was to create a visually clear change within the

middle third of Figure 2.

Figure 2 shows the predictions of the IQM for different values of ρ and θ. Depending on θ, the IQM produces subadditivity, or superadditivity, with minimum unpacking factor occurring when $j(B \wedge A) = j(\bar{B} \wedge A)$ and the maximum unpacking factor occurring when either unpacked judgment is at its maximum value. The maximum unpacking factor is greater for $\rho = .40$ compared to $\rho = .60$.

Decision Process

An agent k submits an order for a bid or ask with equal probability if its money reserve is positive and it owns at least one share. If it owns shares, but has no money, it submits an order for an ask. Otherwise, it submits an order for a bid. When submitting an order for a bid, agent k selects an event $x \in \{e, \bar{e}\}$ with equal probability and sets the value of the bid according to the following rule:

$$\begin{cases} b_{k,x} = a_{x,\min} & a_{x,\min} < j_k(x \\ b_{k,x} \sim \mathrm{D}(\max(0, j_k(x) - \delta), \max(0, j_k(x) - .01)) & \text{Otherwise} \end{cases}$$

where $a_{x,\min}$ is the minimum ask across all agents, $j_k(x)$ is the agent's subjective probability for event x, D is a dis-crete uniform distribution with increments of .01, and $\delta \in \{.01, .02, \ldots, (1 - j_k(x))\}$ defines the variability in the bid or ask. To determine the ask value, the agent considers the consequences of keeping its highest priced share for event x com-pared to selling it for the maximum bid price, denoted $b_{x,\max}$. The expected value of keeping this share is $\mathbb{E}[V_{k,x,\max}]$, and the net payoff for selling it is the difference between the maxi-mum bid and the purchase price of the share: $b_{x,\max} - s_{k,x,\max}$. Letting $z = (b_{x,\max} - s_{k,x,\max}) - \mathbb{E}[V_{k,x,\max}]$, the rule is defined as:

$$\begin{cases} a_{k,x} = b_{\max,x} & z > 0 \\ a_{k,x} \sim \mathrm{D}(\min(1, j_k(x) + .01), \min(1, j_k(x) + \delta)) & \text{Otherwise} \end{cases}.$$

Thus, the agent asks for the maximum bid for event x if it leads to a better payoff; otherwise, it asks for a random value greater than its subjective probability for event x. As a simplifying assumption, the maximum number of orders in each market's order book was limited to one per agent. However, each agent could change any order each day during its turn.

Simulation Results

Figure 3 compares the unpacking factors for the CQM and IQM across time. As expected, the unpacking factor of the CQM varied around a value of 1, indicating a general adherence to additivity with brief violations due to noise. By contrast, the unpacking factor for the IQM reveals a sustained period of subadditivity before shifting to a period of superadditivity upon the agents acquiring new information. For some runs of the IQM, the shift from sub-to-superadditivity is less pronounced, indicating some degree of sensitivity to the distribution of ρ and θ. Overall, these results provide a proof of concept that a shift in perspective in the IQM can produce qualitative deviations from additivity found in prediction markets for the 2024 US elections.

Discussion

Our goal in the present research was to investigate unpacking effects in prediction markets and provide a preliminary model of unpacking effects as a proof of concept. Our investigation of electoral prediction markets uncovered sustained periods of unpacking effects—both in the form of subadditivity and superadditivity. Unpacking effects were primarily of the former variety, and were smaller in magnitude than what is typically found in individual probability judgments. Nonetheless, the unpacking effects were clearly present. As a potential explanation using principles based on quantum cognition, we proposed that unpacking effects could arise from incompatible beliefs of individual actors in the prediction markets. When beliefs are incompatible, the markets are viewed from different psychological perspectives, leading to unpacking effects. If new information prompts individuals to view the markets from a different perspective, this could lead to a period of subadditivity followed by a period of superadditivity, or vice versa. We demonstrated that incompatible representations can produce the qualitative patterns of unpacking effects found in the empirical data, but the unpacking effects disappeared when the model used a compatible rather than incompatible belief representation.

The present research raises many important questions for future research to investigate. One question is why stronger evidence of unpacking effects were found in Sonnemann et al. (2013) and Polymarket in the present study compared to the prediction markets examined by Lee et al. (2009). Some possible explanations include the level of expertise of participants, market dynamics, such as trading volume, procedural differences in the mechanics of the prediction markets, or other unknown moderators. Another question is whether an alternative model might provide better a explanation for unpacking effects. Unlike support theory (Tversky & Koehler, 1994), one advantage of quantum cognition is that it accounts for superadditivity in addition to subadditivity. At this point, however, the evidence for the IQM is tentative, and as we will explain shortly, there are shortcomings associated with this model.

Limitations

Several limitations should be noted. As alluded above, one limitation is that evidence for the IQM's proposed causal mechanism has not been established. Instead, as a proof of concept, we demonstrated that incompatible beliefs can produce some of the qualitative patterns of unpacking effects observed empirically. More work is needed to investigate a possible connection between information consumption and unpacking effects. From a practical perspective, testing the mechanism of the IQM is challenging due to the lack of experimental control and the lack of independent measure of incompatibility. In principle, one could design a closed-system

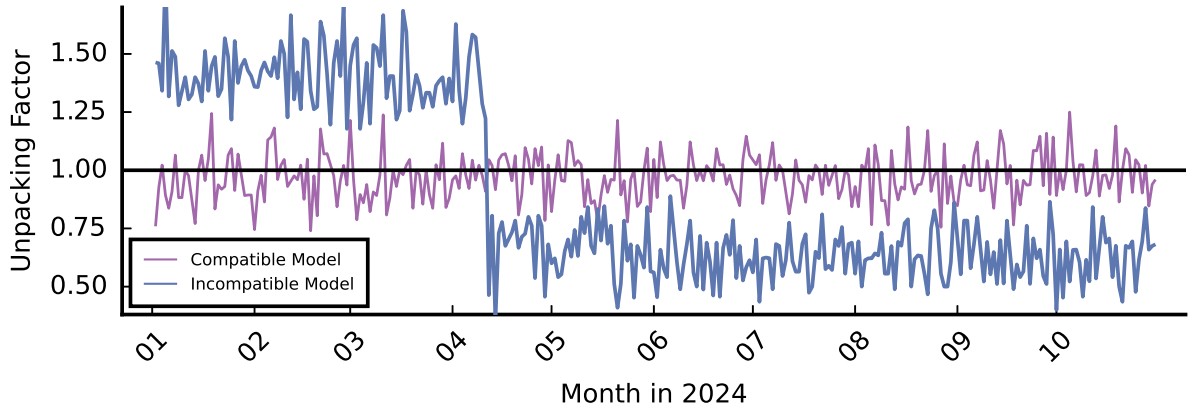


Figure 3: The unpacking factor for the compatible and incompatible quantum models. Values greater than 1 correspond to subadditivity whereas values less than 1 correspond to superadditivity.

prediction market to experimentally test the IQM by attempting to control information consumption and devising an independent measure of compatibility.

Another limitation with the IQM is the sensitivity of unpacking effects to the rotation parameter. As shown in Figure 2, the unpacking factor changes quickly in response to the rotation parameter. Small changes in the distribution over the rotation value may lead to different trajectories for the unpacking factor.

A third limitation is that the IQM assumes that judgments sum to one, even for incompatible events. We found preliminary evidence that the sum of prices across markets forming an entire sample space can be less than or greater than one. Future research might investigate a more recent quantum cognition model called the quantum sampler, which can provide a more comprehensive account of unpacking effects (Huang et al., 2024).

Conclusion

Although prior research has focused primarily on accuracy of prediction markets, coherence may serve as a check on the quality and accuracy of prediction markets, particularly before the outcome is known. We believe that a combination of multi-agent modeling and cognitive modeling approaches can help elucidate the performance of prediction markets, and identify under which conditions they may serve as useful forecasting tools.

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 Department of the Air Force, or any of their subsidiaries or employees. This research was supported through Air Force internal funds. Distribution A: Approved for public release. Case number: AFRL-2025-0498.

References

Arnesen, S., & Strijbis, O. (2015). Accuracy and bias in european prediction markets. *Italian Journal of Applied Statistics*, *2*(25), 123–138.

Berg, J. E., Nelson, F. D., & Rietz, T. A. (2008). Prediction market accuracy in the long run. *International Journal of Forecasting*, *24*(2), 285–300.

Busemeyer, J. R., Pothos, E. M., Franco, R., & Trueblood, J. S. (2011). A quantum theoretical explanation for probability judgment errors. *Psychological Review*, *118*(2), 193–218.

Dana, J., Atanasov, P., Tetlock, P., & Mellers, B. (2019). Are markets more accurate than polls? the surprising informational value of "just asking". *Judgment and Decision Making*, *14*(2), 135–147.

Dreber, A., Pfeiffer, T., Almenberg, J., Isaksson, S., Wilson, B., Chen, Y., ... Johannesson, M. (2015). Using prediction markets to estimate the reproducibility of scientific research. *Proceedings of the National Academy of Sciences*, *112*(50), 15343–15347.

Fiedler, K., Unkelbach, C., & Freytag, P. (2009). On splitting and merging categories: A regression account of subadditivity. *Memory & Cognition*, *37*(4), 383–393.

Franco, R. (2010). Judged probability, unpacking effect and quantum formalism. In *Aaai fall symposium: Quantum informatics for cognitive, social, and semantic processes.*

Huang, J., Busemeyer, J., Ebelt, Z., & Pothos, E. (2024). Bridging the gap between subjective probability and probability judgments: the quantum sequential sampler. *Psychological Review*.

Lee, M. D., Grothe, E., & Steyvers, M. (2009). Conjunction and disjunction fallacies in prediction markets..

Mellers, B., Ungar, L., Baron, J., Ramos, J., Gurcay, B., Fincher, K., ... others (2014). Psychological strategies for winning a geopolitical forecasting tournament. *Psychological science*, *25*(5), 1106–1115.

Page, L., & Clemen, R. T. (2013). Do prediction markets produce well-calibrated probability forecasts? *The Economic Journal*, *123*(568), 491–513.

Polymarket. (2023). *An in-depth overview of polymarket's market making rewards program.* (https://mirror.xyz/polymarket.eth/TOHA3ir5R76bO1vjTrKQclS9k8Dygma53OIzHztJSjk)

Polymarket. (2024). *What is polymarket.* (https://learn.polymarket.com/docs/guides/get-started/what-is-polymarket/)

Sloman, S., Rottenstreich, Y., Wisniewski, E., Hadjichristidis, C., & Fox, C. R. (2004). Typical versus atypical unpacking and superadditive probability judgment. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *30*(3), 573.

Sonnemann, U., Camerer, C. F., Fox, C. R., & Langer, T. (2013). How psychological framing affects economic market prices in the lab and field. *Proceedings of the National Academy of Sciences*, *110*(29), 11779–11784.

Surowiecki, J. (2004). *The wisdom of crowds: why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations*. Doubleday Books.

Tversky, A., & Kahneman, D. (1983). Extensional versus intuitive reasoning: The conjunction fallacy in probability judgment. *Psychological Review*, *90*(4), 293.

Tversky, A., & Koehler, D. J. (1994). Support theory: A nonextensional representation of subjective probability. *Psychological Review*, *101*(4), 547.

Wolfers, J., & Zitzewitz, E. (2004). Prediction markets. *Journal of Economic Perspectives*, *18*(2), 107–126.

Wolfers, J., & Zitzewitz, E. (2006). *Interpreting prediction market prices as probabilities.* National Bureau of Economic Research Cambridge, Mass., USA.

Proceedings of the 23rd International Conference on Cognitive Modelling (ICCM 2025)

Generalizing the priority heuristic to two-stage decision making tasks

Christopher R. Fisher (christopher.fisher.27.ctr@us.af.mil)
Parallax Advanced Research
Beavercreek, OH 45324 USA

Othalia Larue (othalia.larue@parallaxresearch.org)
Parallax Advanced Research
Beavercreek, OH 45324 USA

Kevin Schmidt (kevin.schmidt.15@afrl.af.mil)
Air Force Research Laboratory
Wright Patterson AFB, OH 45433 USA

Abstract

One question in decision making is whether people consider all features before making a decision, or instead consider only a subset of features. One prominent example of the later strategy is the priority heuristic (PH)—a non-compensatory strategy in which features are evaluated sequentially in order of importance until one option is sufficiently better on a given feature dimension. Our primary goal was to test the ability of the PH to generalize to a two-stage decision making task and account for dynamic inconsistency, which occurs when people deviate from their plans for the second stage after observing an outcome from the first stage. We adapted the PH to the two-stage decision task and demonstrated that it predicts dynamic inconsistency under some conditions. However, the predictions were largely at odds with empirical choice patterns—namely, the PH predicted a tendency towards risk aversion, whereas subjects tended to exhibit risk seeking behavior.

Keywords: decision-making; heuristics; dynamic-inconsistency; priority-heuristic

Introduction

One important theoretical debate concerns whether people take into consideration all features of a set of options when making a decision, or instead consider only a subset of features (e.g., Brandstätter et al., 2006; Birnbaum, 2008a). The assumption that people consider all features during decision making is embodied in the notion of *expected utility*, which is ubiquitous throughout economics, cognitive science, and related fields (Brandstätter et al., 2006; Birnbaum, 2008b). Consider the gamble $\mathcal{G} = (x_1, p_1; \ldots; x_n, p_n)$, where x_i is a possible outcome occurring with probability p_i. According to expected utility theory—a theory of rational decision making—one's valuation of the gamble is captured by its expected utility: $\mathrm{EU}(\mathcal{G}) = \sum_i^n p_i u(x_i)$, where utility function $u(x)$ transforms objective values into subjective values. The core assumption of weighting and adding all features also forms the basis of psychologically inspired variants of expected utility theory, including prospect theory (Tversky & Kahneman, 1992), and TAX (Birnbaum, 2008b), which incorporate ideas such as loss aversion and non-linear probability weighting. By contrast, the fast and frugal heuristics approach contends that people make decisions by considering only a subset of important features—in many cases, only a single feature (Gigerenzer & Todd, 1999). Drawing upon bounded rationality, the argument for this approach is that in complex, real world environments it is not feasible to examine and optimally weigh all features to arrive at a decision, nor is it necessary to do so. Instead, people quickly extract a small but diagnostic amount of information to make a decision.

One prominent example of a fast and frugal heuristic is the priority heuristic (PH), which is applied to gambles, such as the one previously described (Brandstätter et al., 2006). According to the PH, people sequentially compare options in terms of minimum outcomes, minimum outcome probabilities, and maximum outcomes, but terminate this evaluation process as soon as one option is sufficiently better in terms of a given feature (see Figure 1). Brandstätter et al. (2006) showed that the PH can account for a variety of phenomena in decision making, including the four fold pattern, the certainty effect, and the Allais Paradox. They also showed that the PH performed at least as well as competing models on several data sets involving binary gambles. Finally, they found some evidence for sequential evaluation of features—namely, the median response times increased with the predicted number of features people considered before making a decision.

Several researchers have subsequently challenged the PH. For example, Birnbaum (2008a) presented evidence that alternative models performed similarly and sometimes better than the PH when the parameters of the alternative models were properly estimated. Moreover, the PH performed poorly compared to alternative models when applied to more diagnostic choice sets, such as gambles with three outcomes instead of two (Birnbaum, 2008a). Along similar lines, Glöckner & Betsch (2008) designed more diagnostic choice sets for testing the PH, and found that prospect theory provided a more accurate account in many cases. Furthermore, Birnbaum (2023) found that empirical data violated a critical property of the PH called interactive independence, according to which an effect of one feature difference is independent of other features which are equal in value across across alternatives. In a reply to various criticisms, Brandstätter et al. (2008) noted that every strategy has a region of poor performance, but argued for an adaptive toolbox of strategies in which the best strategy for the problem is selected.

An alternative strategy for testing the PH is to investigate how well it generalizes to a more complex decision making task. Assuming the PH describes general principles underlying decision making, one would expect it to predict choices on a more complex task without fundamentally changing its core principles. Following this line of reasoning, we tested the ability of the PH to generalize to a two-stage decision task involving a sequence of two related gambles (Barkan & Busemeyer, 2003). In this task, subjects often exhibit *dynamic inconsistency* (DI), a phenomenon whereby planned decisions

for the second stage are subsequently changed after observing the outcome from the first stage. Thus, our goal is to investigate whether the PH can generalize to the two-stage decision making task and account for DI.

Overview

The remainder of this paper is organized as follows. We begin with a description of the two-stage decision task and DI. Next, we describe the PH in more detail and demonstrate how the PH can be extended to the two-stage decision task, and account for DI under some conditions. To test its predictions, we apply the PH to an existing dataset from a two-stage decision task. Finally, we conclude with discussion of key findings, limitations, and pathways for future research.

Two-Stage Decision Task

We tested the generalizability of the PH with a two-stage decision task in which two related gambles are presented sequentially (Barkan & Busemeyer, 2003; Busemeyer et al., 2015). In the first stage, participants do *not* make a decision. Instead, an outcome $x_{i,1} \in \{x_G, x_L\}$ is sampled from a risky, binary gamble $\mathcal{R}_1 = (x_G, .50; x_L, .50)$ and then presented to the participant, where $x_G \geq 0$ is a gain, $x_L \leq 0$ is a loss, and the index of $\mathcal{R}$ corresponds to stage. Henceforth, we suppress the stage index in $x_{i,1}$ for ease of notation, as we do not discuss outcomes from stage 2. In the second stage, participants decide between a risky option $\mathcal{R}_2 = \mathcal{R}_1$, and a safe option $\mathcal{S}_2 = (0, 1.0)$.

Each participant performed the two-stage decision task in two conditions: a *planned decision* condition followed by a *final decision* condition (Barkan & Busemeyer, 2003). In the planned decision condition, participants made a plan for choosing between $\mathcal{R}_2$ and $\mathcal{S}_2$ contingent on a hypothetical outcomes x_L and x_G from stage 1. Next, in the final decision condition, participants observed an outcome x_i from stage 1 and decide whether to commit to the corresponding planned decision for stage 2, or switch to the other option. As shown in Table 1, there were 16 unique gambles plus 1 practice gamble (Barkan & Busemeyer, 2003). The experienced outcome in stage 1 was x_L for nine gambles, and was x_G for the remaining eight. Each subject made planned and final decisions for each gamble twice, except for the practice gamble.

Dynamic Inconsistency

One of the primary goals of the two-stage decision task is to examine whether participants change their planned decisions after observing the outcome from the first stage in the final decision condition. The phenomenon whereby planned decisions differ from final decisions is termed *dynamic inconsistency* (DI; Busemeyer et al., 2015). Assuming a closed system in which the participant knows all relevant information pertaining to the decision, expected utility theory predicts that subjects make the same decision in both conditions (Johnson & Busemeyer, 2001). The solution for identifying the optimal decision in multi-stage decision making is obtained through a process called backward induction (Johnson & Busemeyer, 2001). Backward induction involves constructing a decision tree spanning the decision stages, where nodes represent decision points and possible outcomes, and edges represent possible pathways. The optional solution is obtained by pruning inferior options, starting at the last stage, and moving successively up the decision tree until only one sequence of decisions remains. Importantly, backward induction is predicated on the assumption of *dynamic consistency*—the notion that preferences are stable across time. Consequentially, backward induction entails that planned and final decisions must be the same.

Priority Heuristic

As shown in Figure 1, the PH assumes decisions are made by comparing the features between options in descending order of importance, and selecting an option and soon as its comparative advantage exceeds an aspiration level on a given feature (Brandstätter et al., 2006). Unlike models based on expected utility theory, the PH is a non-compensatory strategy, meaning as soon as an option is sufficiently better on one feature dimension, the superior option with respect to this feature is selected without regard for the other features. In principle, a decision can be made by considering only the first feature.

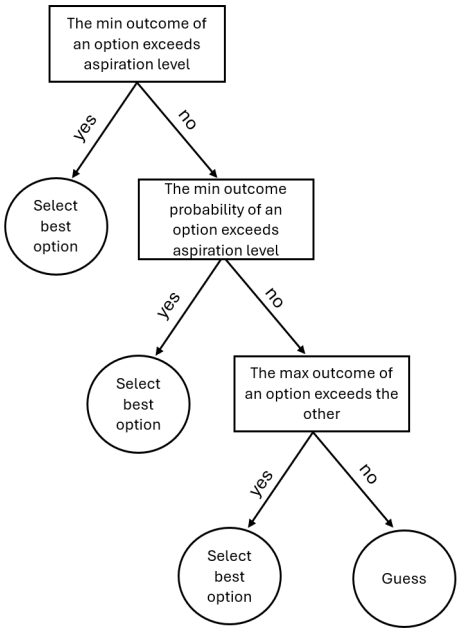


Figure 1: An illustration of the priority heuristic. Starting at the top, options are compared in terms of features ranked in descending order of importance. For a given feature, an option is selected if has the best value above an aspiration level. Otherwise, the options are compared in terms of the next most important feature.

The PH uses an identity function to assess the utility of a feature: $U(x) = x$. Thus, the valuation of features is assumed to be objective. According to the PH, features of gambles are compared in descending order of importance:

(1) the minimum outcome, (2) the probability of the minimum outcome, and (3) the maximum outcome. In the gamble $\mathcal{G} = (5, .90; 10, .10)$, those features correspond to 5, .90, and 10, respectively. The definition of minimum outcome and maximum outcome differs somewhat from the conventional, mathematical definition of minimum and maximum. Instead, both depend on the value of the two outcomes under consideration (Brandstätter et al., 2006; Rieskamp, 2008). General definitions that apply to gambles with any outcome, including mixed gains and losses, have not described formally. Here, we formalize our interpretation based on verbal descriptions in prior research (Brandstätter et al., 2006; Rieskamp, 2008). The minimum outcome of a binary gamble is defined formally with the function:

$$f_{\min}(x,y) = \begin{cases} \max(x,y) \text{ if } x \leq 0, y \leq 0 \\ \min(x,y) \text{ otherwise} \end{cases} . \quad (1)$$

Thus, for non-positive outcomes, the goal is to select the minimum loss. The corresponding maximum outcome of a binary gamble is defined according to the function:

$$f_{\max}(x,y) = \begin{cases} \min(x,y) \text{ if } x \leq 0, y \leq 0 \\ \max(x,y) \text{ otherwise} \end{cases} . \quad (2)$$

For simplicity, we define $x_{\min,j} = f_{\min}(x_j, y_j)$ and $x_{\max,j} = f_{\max}(x_j, y_j)$ as the minimum outcome and maximum outcome respectively, where $j \in \{\text{risky}, \text{safe}\}$ indexes the decision option. The probability of the minimum outcome for option j is denoted $p_{\min,j}$.

The aspiration level is the minimum difference between the feature values of two options required to make a decision. If the aspiration level is met or exceeded, the option with the superior feature value is selected (Brandstätter et al., 2006). Otherwise, the options must be compared with respect to the next most important feature. The aspiration level for the minimum outcome is defined relative to other outcomes, rather than an absolute value. In particular, the aspiration level is defined as 10% of the outcome with largest magnitude across all options. The outcome with the largest magnitude is defined as:

$$x_{\max} = \max(\{|x_{i,j}|\}_{i \in I_i, j \in I_j}), \quad (3)$$

where I_i, and I_j are index sets for outcomes and options, respectively. The quantity $x_{\max}$ is often rounded to the nearest prominent number, which is defined as a power of 10, or half or twice a power of ten. For simplicity, we omit this step without significantly altering predictions. For the probability of the minimum outcome, the aspiration level is 10% of the maximum of the probability scale, which is $.10 \cdot 1 = .10$.

Putting the pieces above together, the decision rule for comparing minimum outcomes is:

$$\begin{cases} \text{choose safe if } (x_{\min,\text{safe}} - x_{\min,\text{risky}}) \geq .10 \cdot x_{\max} \\ \text{choose risky if } (x_{\min,\text{risky}} - x_{\min,\text{safe}}) \geq .10 \cdot x_{\max} \\ \text{check next feature otherwise} \end{cases} . \quad (4)$$

Similarly, the decision rule for comparing the minimum outcome probabilities is:

$$\begin{cases} \text{choose safe if } (p_{\min,\text{risky}} - p_{\min,\text{safe}}) \geq .10 \\ \text{choose risky if } (p_{\min,\text{safe}} - p_{\min,\text{risky}}) \geq .10 \\ \text{check next feature otherwise} \end{cases} . \quad (5)$$

Note that the goal above is to minimize the probability of the worst outcome. Finally, the decision rule for comparing maximum outcomes is:

$$\begin{cases} \text{choose safe if } x_{\max,\text{safe}} > x_{\max,\text{risky}} \\ \text{choose risky if } x_{\max,\text{risky}} > x_{\max,\text{safe}} \\ \text{guess with equal probability otherwise} \end{cases} . \quad (6)$$

To see how the PH works, consider the following choice between a risky option $\mathcal{R} = (10, .50; -10, .50)$ and a safe option $\mathcal{S} = (0, 1)$. After substituting the relevant values into Equations 1 and 3, we obtain $x_{\min,\text{risky}} = -10$, $x_{\min,\text{safe}} = 0$, and $x_{\max} = 10$. The PH predicts that one should select the safe option based on the minimum outcomes because

$$0 - (-10) \geq .10 \cdot 10$$
$$10 \geq 1.$$

It should be noted that in its current formulation the PH cannot account for DI, owing to the fact that the decision process does not vary with experimental condition. Next, we turn to a proposed solution for producing DI with the PH, which entails a modification to the valuation process while maintaining its core decision logic.

Extending the Priority Heuristic

In this section, we explain how the PH can be extended to account for DI in the two-stage decision making task using *reference point dependent valuation*. This concept formed the basis for an extension of prospect theory called the R model, which was used to account for DI (Barkan & Busemeyer, 2003) and similar effects (Tversky & Shafir, 1992). As the name implies, reference point dependent valuation allows one to compare options using a reference point that depends on experimental condition.

Reference point dependent valuation works as follows: In the planned decision condition, the risky option $\mathcal{S}_2$ and the safe option $\mathcal{R}_2$ are evaluated without consideration of the hypothetical outcome x_i in the first stage. Consequentially, the valuations of the outcomes are $\{x_G, x_L\}$ for $\mathcal{R}_2$, and $\{0\}$ for $\mathcal{S}_2$. By contrast, in the final decision condition, the realized outcome x_i from the first stage is incorporated into the valuation of the options, producing $\{x_i + x_G, x_i + x_L\}$ for $\mathcal{R}_2$, and $\{x_i\}$ for $\mathcal{S}_2$. One rationale for omitting the hypothetical outcome in the planned decision condition is that it may have lower salience or emotional impact compared to experienced outcomes, causing the decision maker to completely discount the hypothetical outcome. Henceforth, we will refer to this extension as the reference-point priority heuristic (RPPH).

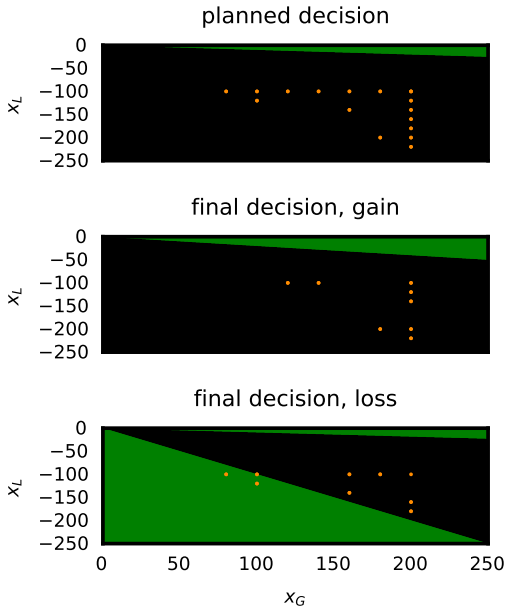


Figure 2: Predictions of the reference point priority heuristic (RPPH) in each condition. Notes: Actions are color coded as black: select safe option, green: select risky option. Orange dots represent gamble outcomes used in (Barkan & Busemeyer, 2003).

Predictions

The predictions of the RPPH can be obtained by substituting the reference point dependent values into Equations 1, 2, and 3, and substituting the resulting values into the decision rules defined in Equations 4, 5, and 6. After some algebraic manipulation, the predictions for each latent, feature comparison stage can be represented as a system of linear inequalities within $x_G \times x_L$ stimulus space. The choice predictions for observed decisions are defined by the union of the regions across the latent feature comparison stages.

The qualitative predictions for DI can be gleaned by comparing the planned decisions to the final decisions in Figure 2. One key insight is that the RPPH can produce DI in some regions of the prediction space. When the first stage outcome is a gain x_G, the RPPH predicts DI for a narrow band located at the top right of the prediction space. By contrast, when the first stage outcome is a loss x_L, the RPPH predicts DI in a much larger region, defined by $|x_L| \geq x_G$. For the gambles used in Barkan & Busemeyer (2003), the RPPH does not predict DI for outcome x_G. Instead, it predicts that people will select the safe option for both planned and final decisions. However, for three of nine gambles, the RPPH does predict DI when the first stage outcome is x_L, such that people shift from the safe option in the planned condition to the risky option in the final decision. In the remaining gambles, the RPPH predicts the selection of the safe option for the planned and final decision.

True and Error Model Analysis

One challenge in evaluating the RPPH lack of error model to account for various sources of stochastic behavior, including lapses in attention, misreading information, and selecting the incorrect response key. Defining an error model is important for two reasons: (1) human behavior is inherently stochastic, and (2) standard statistical tests may yield misleading results because they do not properly account for true preferences and errors (Birnbaum, 2023).

We used a True and Error Model (TEM; Birnbaum, 2023) to disentangle true preferences from errors while making minimal assumptions about the error distribution. The two-stage decision task in Barkan & Busemeyer (2003) meets the following requirements of TEM: (1) using two choice sets (e.g., planned vs. final decision), and (2) repeating each choice set at least twice. For each unique group of four choice sets, the set of possible responses consists of $2^4 = 16$ tuples: $\{(R_{p,1}, R_{f,1}, R_{p,2}, R_{f,2}), \ldots, (S_{p,1}, S_{f,1}, S_{p,2}, S_{f,2})\}$, where S and R indicate the selection of the safe and risky gambles, respectively, sub-scripts p, and f index planned and final decision conditions, and the second subscripts index replication. A TEM can be conceptualized as a multinomial processing tree, consisting of nodes for true preference states and error states, and branches which represent transitions between nodes, each occurring with some probability. This tree like structure defines multiple pathways to a response, each traversing through between different true preference states and error states. In the TEM, the true preference state space is defined by $\{(R_p, R_f), (R_p, S_f), (S_p, R_f), (S_p, S_f)\}$, and parameters $p_{\mathrm{R_pR_f}}, p_{\mathrm{R_pS_f}}, p_{\mathrm{S_pR_f}}, p_{\mathrm{S_pS_f}}$ represent the joint probability distribution over true preference states. In addition, the TEM has four error parameters: $\varepsilon_{\mathrm{R}_p}, \varepsilon_{\mathrm{R}_f}, \varepsilon_{\mathrm{S}_p}, \varepsilon_{\mathrm{S}_f} \leq .50$, where, for example, $\varepsilon_{\mathrm{R}_p}$ is the probability of erroneously selecting $\mathcal{R}$ given a true preference for $\mathcal{S}$ in the planned decision condition.

Consider a person who selects $\mathcal{R}$ in both choice sets and both replicates: $(R_{p,1}, R_{f,1}, R_{p,2}, R_{f,2})$. This response pattern could have been generated by any of the four preference states, each with a different error pattern, i.e., path in the tree. For example, the path probability for preference state (R_p, R_f) is $p_{\mathrm{R_pR_f}} \cdot (1-\varepsilon_{\mathrm{S}_p})^2 \cdot (1-\varepsilon_{\mathrm{S}_f})^2$. The paths for the other three preference states can be defined using similar logic. All four paths sum to produce the marginal probability of the example response pattern above.

Results

We used Bayes factors to test the predictions of RPPH. A Bayes factor is the ratio of marginal likelihoods between two competing models, $\mathcal{M}_i$ and $\mathcal{M}_j$:

$$\mathrm{BF}_{i,j} = \frac{f(\mathbf{Y} \mid \mathcal{M}_i)}{f(\mathbf{Y} \mid \mathcal{M}_j)}, \tag{7}$$

where f is a probability density function, $\mathbf{Y} = [y_1, y_2, \ldots, y_n]$ is a vector of data, and the marginal likelihood of $\mathcal{M}_i$ is given

by:

$$f(\mathbf{Y} \mid \mathcal{M}_i) = \int_{\theta \in \Theta_i} f(\mathbf{Y} \mid \theta, \mathcal{M}_i)\pi(\theta \mid \mathcal{M}_i)\mathbf{d}\theta, \qquad (8)$$

where π is the prior density over parameter vector $\theta \in \Theta_i$, and Θ_i is the parameter space for $\mathcal{M}_i$. The Bayes factor can be interpreted as the relative probability of the data under $\mathcal{M}_i$ vs. under $\mathcal{M}_j$. Alternatively, it can be interpreted as the factor by which prior model odds must be updated to obtain the posterior model odds. Bayes factors have two attractive features: (1) they naturally account for various sources of model flexibility, including functional form and dimensionality, and (2) they can be used even when there is disagreement over the prior model odds.

Table 1: A comparison of the predictions for the RPPH and data for each gamble from Barkan & Busemeyer (2003). Columns x_G and x_L correspond to possible gains and losses of each gamble. Outcome refers to the experienced outcome from stage 1. Predictions are listed under RPPH, where S = select safe option, R = select risky option, and the positions within each pair correspond to planned decision and final decision, respectively. Bayes factors compare RPPH relative to true and error model with a small DI effect.

x_G	x_L	Outcome	RPPH	$\log_{10}$(BF)
200	-200	x_G	SS	-12
120	-100	x_G	SS	-27
180	-200	x_G	SS	-15
200	-120	x_G	SS	-30
200	-220	x_G	SS	-16
200	-100	x_G	SS	-36
140	-100	x_G	SS	-24
200	-140	x_G	SS	-18
200	-100	x_L	SS	NA
160	-140	x_L	SS	-17
180	-100	x_L	SS	-36
100	-120	x_L	SR	-8
200	-160	x_L	SS	-28
160	-100	x_L	SS	-25
200	-180	x_L	SS	-21
80	-100	x_L	SR	-16
100	-100	x_L	SR	-15

We compared $\mathcal{M}_{\text{RPPH}}$ to an alternative model, $\mathcal{M}_{\text{DI}}$, which predicts a small DI effect. The prior distributions for this model are $\mathbf{p} \sim \text{Dirichlet}([5,1,1,5])$ for preference state probabilities and $\varepsilon \sim \text{uniform}(0,.50)$ for error probabilities. The priors for $\mathcal{M}_{\text{RPPH}}$ are the same as $\mathcal{M}_{\text{DI}}$, except constraints are imposed on $\mathbf{p}$ based on trial specific predictions. As an example, consider the gamble listed in the first row of Table 1 where the predicted preference is for the safe option in both conditions. Thus, the parameter $p_{S_pS_f} = 1$, with the remaining preference state probabilities set to zero. Hence, any deviation from selecting only the safe option must be attributed exclusively to error.

Table 1 shows Bayes factors comparing $\mathcal{M}_{\text{RPPH}}$ to $\mathcal{M}_{\text{DI}}$ in $\log_{10}$ units. A positive value indicates support for $\mathcal{M}_{\text{RPPH}}$, whereas a negative value indicates support for $\mathcal{M}_{\text{DI}}$. A value of zero indicates both models are equally supported by the data. Across the different gambles, the $\log_{10}$ Bayes factors are strongly negative, providing clear evidence against $\mathcal{M}_{\text{RPPH}}$. One potential issue with Bayes factors is that they can be sensitive to the choice of priors over parameters. As a robustness check, we compared to an alternative with a uniform prior over true preference states: $\mathbf{p} \sim \text{Dirichlet}([1,1,1,1])$, but the results were qualitatively similar.

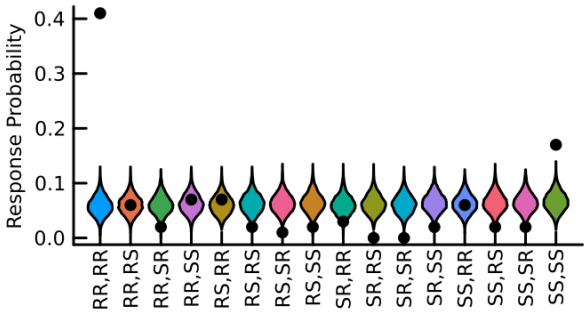


Figure 3: Posterior predictive distributions of $\mathcal{M}_{\text{RPPH}}$ for gamble 2 in Table 1. Response patterns are labeled on the x-axis, and data are represented by black solid dots.

To understand why $\mathcal{M}_{\text{RPPH}}$ performed poorly, we compared the data to its posterior predictive distribution. Figure 3 provides a representative example in which the predictions were approximately uniform, and failed to account for several aspects of the empirical data. For example, it underestimated subjects' tendency to consistently select $\mathcal{R}$ or consistently select $\mathcal{S}$.

Discussion

In this research, we extended the PH to a two-stage decision making task to evaluate its generalizability to more complex decisions. In service of this goal, we incorporated a previously reported reference point dependent valuation process into the PH to account for DI, while maintaining the core decision logic of the PH. We termed the resulting model the RPPH. Our analysis of the RPPH revealed that it predicts DI under some conditions, including those in the experimental design used to test the RPPH (see Barkan & Busemeyer, 2003). Examination of the RPPH's prediction space shows that it predicts DI under more conditions when the first stage outcome is a loss (x_L).

Although the RPPH can produce DI in principle, its predictions were frequently at odds with the choice patterns observed in the experiment conducted by Barkan & Busemeyer (2003). We evaluated RPPH using a TEM to distinguish true preferences from a errors, using minimal assumptions regarding the source and distribution of errors. Nonetheless, the

discrepancy between the data and the model predictions were quite large. In many cases, the RHHP predicted a preference for the safe options, but subjects selected the risky option at a might higher rate than could be explained by high error probabilities.

Limitations

Several limitations and caveats are worth noting. Although we did not alter the decision logic of the PH when extending it to the two-stage decision task, we introduced a reference point dependent valuation process which is not part of its core theory. A different mechanism might yield more accurate predictions. Here, we opted for a mechanism that seemed psychologically plausible, and showed some success in prior research (e.g., Barkan & Busemeyer, 2003). Another consideration is that the PH is only applicable to difficult choices, commonly defined as options with a ratio of expected values less than or equal to two. This constraint is satisfied for final decisions, but the ratio is undefined for planned decision because the expected value of one option is zero. One could argue that ratio of expected values is problematic because its not defined when one expected value is zero, and it does not consider the variance of the payoff distributions. When considering $\frac{\mathbb{E}(\mathcal{R})-\mathbb{E}(\mathcal{S})}{\sqrt{\text{Var}(\mathcal{R})+\text{Var}(\mathcal{S})}}$ as alternative measure of decision difficulty, the choice sets used in Table 1 have a low signal-to-noise ratio, indicating high decision difficulty.

A critic could also argue that the RPPH is overly restrictive because it does not allow individual differences in various aspects of the model, such as aspiration levels. In future work, researchers could examine a more flexible variant to the RPPH based on the stochastic PH (Rieskamp, 2008), which relaxes the assumptions that aspiration levels and feature evaluation order are fixed.

Conclusion

The question of which features people consider during decision making, and how—if at all—they might be integrated into a singular summary value (e.g., expected value) has long intrigued researchers and spurred theoretical debate (e.g., Gigerenzer & Todd, 1999; Birnbaum, 2008a; Brandstätter et al., 2008; Glöckner & Betsch, 2008). In our contribution to this debate, we assessed the ability of the PH to generalize to a more complex decision task compared to simpler decision tasks used in previous research. With some caveats, our findings cast doubt on the non-compensatory decision processes specified by the PH, and add to a list of challenges to the PH. Our research provides an entry point for future research to analyze the PH in more complex tasks and attempt to reach stronger conclusions.

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 Department of the Air Force, or any of their subsidiaries or employees. This research was supported through Air Force internal funds. Distribution A: Approved for public release. Case number: AFRL-2025-0499.

References

Barkan, R., & Busemeyer, J. R. (2003). Modeling dynamic inconsistency with a changing reference point. *Journal of Behavioral Decision Making*, *16*(4), 235–255.

Birnbaum, M. H. (2008a). Evaluation of the priority heuristic as a descriptive model of risky decision making: Comment on brandstätter, gigerenzer, and hertwig (2006). *Psychological Review*, *115*(1), 253–260.

Birnbaum, M. H. (2008b). New paradoxes of risky decision making. *Psychological Review*, *115*(2), 463.

Birnbaum, M. H. (2023). True and error analysis instead of test of correlated proportions: Can we save lexicographic semiorder models with error theory? *Psychological Methods*.

Brandstätter, E., Gigerenzer, G., & Hertwig, R. (2006). The priority heuristic: making choices without trade-offs. *Psychological Review*, *113*(2), 409.

Brandstätter, E., Gigerenzer, G., & Hertwig, R. (2008). Risky choice with heuristics: reply to birnbaum (2008), johnson, schulte-mecklenbeck, and willemsen (2008), and rieger and wang (2008). *Psychological Review*, *115*(1), 281–289.

Busemeyer, J. R., Wang, Z., & Shiffrin, R. M. (2015). Bayesian model comparison favors quantum over standard decision theory account of dynamic inconsistency. *Decision*, *2*(1), 1.

Gigerenzer, G., & Todd, P. M. (1999). Fast and frugal heuristics: The adaptive toolbox. In *Simple heuristics that make us smart* (pp. 3–34). Oxford University Press.

Glöckner, A., & Betsch, T. (2008). Do people make decisions under risk based on ignorance? an empirical test of the priority heuristic against cumulative prospect theory. *Organizational Behavior and Human Decision Processes*, *107*(1), 75–95.

Johnson, J. G., & Busemeyer, J. R. (2001). Multiple-stage decision-making: The effect of planning horizon length on dynamic consistency. *Theory and Decision*, *51*, 217–246.

Rieskamp, J. (2008). The probabilistic nature of preferential choice. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *34*(6), 1446.

Tversky, A., & Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and Uncertainty*, *5*, 297–323.

Tversky, A., & Shafir, E. (1992). The disjunction effect in choice under uncertainty. *Psychological Science*, *3*(5), 305–310.

Proceedings of the 23rd International Conference on Cognitive Modelling (ICCM 2025)

Examining the Sensitivity of Prediction Markets to Price Manipulation and Disinformation: A Simulation Study

Christopher R. Fisher (christopher.fisher.27.ctr@us.af.mil)
Parallax Advanced Research
Beavercreek, OH 45324 USA

Taylor Curley(taylor.curley@us.af.mil)
Air Force Research Laboratory
Wright Patterson AFB, OH 45433 USA

Abstract

Although prediction markets can be used as a forecasting tool, concerns regarding their susceptibility to various forms of price manipulation have limited their application. Using agent-based modeling, we investigate the effectiveness of two price manipulation strategies under a wide range of conditions: (1) a bad actor who inflates the market price with high bids, and (2) the spread of disinformation among market participants. Our results indicate that a bad actor can exert a wide range of price manipulation effects, depending on budget constraints and how quickly other agents can clear inflated bids. However, the equilibrium price was reestablished by the end of the trading period in most cases. Our results indicate that a social network with a power law degree distribution efficiently spreads disinformation. Disinformation could have a persistent effect unless it creates an incentive for new agents to enter the market or fact verification is not costly. We conclude with a discussion of limitations.

Keywords: disinformation; prediction markets; forecasting; decision-making; agent-based modeling

Introduction

A prediction market (PM) is a forecasting tool which leverages the profit motive to distill heterogeneous knowledge distributed across a large number of individuals into a market price. Importantly, the market price is often interpreted as a crowd-sourced probability estimate of a future event (Wolfers & Zitzewitz, 2006). In a PM, much like a stock market, participants buy and sell shares. However, a key difference compared to a stock market is that a share represents a contract that pays 1 dollar if a future event occurs and nothing otherwise. PMs can be used to forecast a wide variety of events, including geopolitics, sporting outcomes, and project deadlines—provided that the events are well-defined (e.g., Page & Clemen, 2013).

Generally speaking, PMs are useful—albeit imperfect—forecasting tools. For U.S. elections between 1988 and 2004, PMs were more accurate than polling over extended time horizons (Berg et al., 2008). Page & Clemen (2013) examined the calibration of PMs for a wide variety of elections and sporting events, and found evidence for the favorite-longshot bias, defined as the tendency for low probability events to be over-priced and high probability events to be under-priced. However, as the time horizon shortened, this bias gradually diminished. Some evidence indicates that combining PMs with de-biased individual judgments leads to some improvement in accuracy, suggesting PMs do not aggregate information in a completely efficient manner (Dana et al., 2019).

Although PMs show promise as forecasting tools, concern regarding their susceptibility to manipulation has limited their adoption to some degree (e.g., Wolfers & Zitzewitz, 2004). One concern is that price manipulation could sabotage decision makers or policy makers who incorporate PMs into their decision-making process. In elections, price manipulation could amplify a potential bandwagon effect, whereby voters increase their support for a candidate who appears to be favored to win (see Morton et al., 2015, for an example involving exit polls). Even the perceived risk of manipulation—whether founded or not—could undermine trust in an otherwise useful forecasting tool.

One strategy for price manipulation involves a well-funded bad actor who purchases a large quantities of shares above or below the market price depending on the desired effect. A common argument for the robustness of PMs against this form of manipulation is that they have several built-in safeguards—namely, price manipulation is costly for the manipulator, knowledge is decentralized, and price manipulation creates an opportunity for informed individuals to exploit a mis-priced asset (e.g., Hayek, 1945). The empirical evidence for price manipulation in PMs is somewhat mixed. Evidence of successful price manipulation was found in a PM for a Berlin state election (Hansen et al., 2004). However, a manipulation attempt in horse race betting failed to have a discernible effect on prices (Camerer, 1998). In a PM experiment investigating price manipulation, a subset of participants were given an additional incentive to increase the market price (Hanson et al., 2006). Participants were unaware of who had the additional incentive, but the presence and direction of the incentive was disclosed. No evidence of price manipulation was found under these conditions. One remaining question is whether the manipulation attempt would have been effective if participants were *not* informed of the incentive and its likely effect on price. In a PM experiment where manipulators knew the true price and were paid based on the amount of money forecasters incorrectly invested, Deck et al. (2013) found evidence of successful price manipulation.

Another price manipulation strategy is to indirectly influence prices using disinformation designed to bias individual beliefs in a desired direction. In principle, disinformation could have pervasive effects for two reasons. One reason is that disinformation has a natural tendency to propagate through social networks, thus extending the reach of its effect. Moreover, social media could signal boost disinformation and reinforce its message with echo chambers. Another reason is that prior research has found that the effect of mis/disinformation can be resistant to corrective information, a phenomenon known as the continued influence effect (Johnson & Seifert, 1994). To the best of our knowledge, little

research has investigated the potential role of misinformation in PMs.

Overall, the evidence for price manipulation in PMs does not paint a clear story, leaving the boundary conditions and magnitude of price manipulation poorly understood. We believe that agent-based modeling can play an important role in elucidating these questions and spurring future empirical research. Agent-based modeling allows one to develop a large computational testbed for investigating price manipulation under a wide range of assumptions pertaining to manipulation strategy, decision making, and mitigation strategy. This approach has two advantages: (1) it allows one to examine the robustness of PMs to price manipulation under alternative assumptions, and (2) it sharpens research questions and serves as a guide for developing informative experiments. For these reasons, we employ agent-based modeling to understand the two price manipulation strategies described above. In the first simulation, we investigate how the budget of a single manipulator and the trading rate of other agents affects price manipulation. In the second simulation, we investigate how the strength of disinformation and interconnectedness of a social network affects price manipulation.

Simulation 1: Price Manipulation

Our goal in Simulation 1 is to investigate how two factors affect a bad actor's ability to manipulate prices: (1) the size of the bad actor's budget, and (2) the rate at which other agents sell inflated shares. Our general expectation is that increasing the bad actor's budget will allow it to inflate the price for a longer duration, and increasing the trade rate of the other agents will mitigate price manipulation. Going beyond these qualitative relationships, we will use agent-based modeling to elucidate the magnitude of price manipulation effects. In this simulation, we introduce two types of agents: a set of *standard agents* who trade shares based on their subjective beliefs, and a single *manipulator* agent, who expends its entire budget in attempt to inflate the market price. In what follows, we describe the mechanics of the PM, and the rules governing the decision making of both agent types.

Prediction Market

We used a continuous double auction PM whereby orders for bids and asks are tabulated in an order book which is visible to all participating agents. As detailed below, a transaction occurs when an equilibrium is achieved (e.g., bid and ask match). Non-matching orders remain in the order book until a match is found. In the PM, agents trade shares representing a payment contract for the occurrence of a future binary event $x \in \{e, \bar{e}\}$, where e is an event (e.g., Federal Reserve will raise interest rates) and $\bar{e}$ is the complementary event (e.g., Federal reserve does *not* raise interest rates). A share for event x pays 1 dollar if x occurs and 0 dollars otherwise. If an agent purchases a share for event x at the price of $s_x \in [0,1]$, the net payout is $1 - s_x$ if x occurs and $-s_x$ otherwise. At time t, the market price is $p_{x,t} \in [0,1]$, which represents a dynamic, crowd-sourced probability estimate for the occurrence of event x.

In what follows, we provide a detailed explanation of transaction conditions governing trade activity in the PM. First, define b_e and a_e as the bid price and ask price for event e, respectively. For binary events, a bid (ask) is equivalent to 1 minus an ask (bid) for the complementary event: $b_e = 1 - a_{\bar{e}}$. Given these definitions, a transaction occurs under three conditions: (1) if $b_{i,e} = a_{k,e}$, agent i pays agent $k \neq i$ the amount of $b_{i,e}$ in exchange for one share; (2) if $b_{i,e} + b_{k,\bar{e}} = 1$, a share for e is created and given to agent i for the amount of $b_{i,e}$, and likewise a share for $\bar{e}$ is created and given to agent k for the amount of $b_{k,\bar{e}}$; and (3) if $a_{i,e} + a_{k,\bar{e}} = 1$, agent i receives $a_{i,e}$ and relinquishes 1 share for event e, and agent k receives $a_{k,\bar{e}}$ and relinquishes 1 share for event $\bar{e}$.

Standard Agents

Standard agents buy and sell shares in the PM using simple rules based on their subjective probability estimates. When making a bid, the agent considers the best current offer, which is the minimum ask price. If the minimum ask will yield a positive expected net payoff, the agent accepts it. Otherwise, the agent submits to the order book a random bid less than its subjective probability. To explain the bid rule more precisely, we define $j_{k,x}$ as agent k's subjective probability of event x, and the random variable V_x as the net payoff of a share for event x purchased for s_x. Thus, the subjective expected value of a share is given by $\mathbb{E}[V_x] = j_{k,x} \cdot [1 - s_x] - [1 - j_{k,x}] \cdot s_x = j_{k,x} - s_x$, which is positive if $s_x < j_{k,x}$. With these terms defined, the bid rule is given formally by:

$$\begin{cases} b_{k,x} = a_{\min,x} & a_{\min,x} < j_{x,k} \\ b_{k,x} \sim \mathrm{D}(\max(0, j_{k,x} - \delta), \max(0, j_{k,x} - .01)) & \text{Otherwise} \end{cases},$$

where $a_{\min,x}$ is the minimum ask in the order book, and D is a discrete uniform distribution with increments of .01, and $\delta \in \{.01, .02, \ldots, (1 - j_{k,x})\}$ defines the variability in the bid or ask price.

When determining the ask price, a standard agent compares two options: (1) the subjective expected value of keeping its highest priced share, and (2) selling that share for the maximum bid price in the order book. The expected value of keeping the max priced share is $\mathbb{E}[V_{k,\max,x}]$, and the net payoff for selling it at the maximum bid price is the difference between the maximum bid and the purchase price of the share: $b_{\max,x} - s_{k,\max,x}$. Letting $z = (b_{\max,x} - s_{k,\max,x}) - \mathbb{E}[V_{k,\max,x}]$, the rule is defined as:

$$\begin{cases} a_{k,x} = b_{\max,x} & z > 0 \\ a_{k,x} \sim \mathrm{D}(\min(1, j_{k,x}) + .01), \min(1, j_{k,x} + \delta)) & \text{Otherwise} \end{cases}.$$

A standard agent k submits an order for a bid or ask with equal probability if its money reserve is positive and it owns at least one share. If it owns shares, but has no money in reserve, it submits an order for an ask. Otherwise, it submits an order for a bid.

Manipulator

Unlike standard agents, the goal of the manipulator agent is to inflate the market price by purchasing as many shares as possible at the maximum possible price. This process proceeds first by purchasing as many asks as possible in the order book. Next, with its remaining budget, the manipulator agent submits as many bids as possible for 1 dollar. Under most conditions, this will cause the market price to quickly escalate to 1 dollar. Additionally, the manipulator holds its shares in an effort to prolong the duration of the price manipulation.

Design Parameters

Simulation 1 consisted of 500 standard agents, each initialized with a budget of 30 dollars. Each standard agent submitted an order (bid or ask) once per day throughout a trading period of 50 days. Standard agents submitted orders in a different random sequential order each day. Additionally, each standard agent removed all of its outstanding orders from the order book before submitting a new order. On day 5, the manipulator expended its budget to purchase all asks in the order book and used any remaining budget to submit as many bids as possible for 1 dollar each.

For each standard agent k, a subjective probability was sampled from a beta distribution as follows: $j_{k,e} \sim \text{beta}(\mu \cdot \eta, (1-\mu) \cdot \eta)$, using $\mu = .30$ as the expected subjective probability across standard agents and $\eta = 20$ as the precision parameter. Additionally, we set the bid/ask range parameter to $\delta = .03$. We systematically varied the budget of the manipulator as a percentage of the total budget across standard agents ($\$30 \cdot 500 = \$15,000$), and the trade rate, defined as the maximum number of shares each standard agent can buy or sell each day (subject to budget constraints). Accordingly, the ratio of these quantities determines the duration of a price manipulation. The manipulator's budget % ranged from from 0% to 50% in increments of 5%, and the trade rate ranged from 2 to 20 in increments of 2.

On each model run, we quantified the degree of price manipulation with two metrics: (1) the overall effect of price manipulation using the root mean squared error (RMSE) between μ and market prices across time: $\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\mu - p_{e,t})^2}$, and (2) the effect of price manipulation on the final price using RMSE evaluated at time T. For $\mu = .30$, the theoretical range of RMSE is $[0, .70]$. We repeated the model simulation 10 times for each combination of manipulator budget % and trade rate, and averaged both metrics across repetitions.

Simulation Results

Figure 1 illustrates price manipulation for a single simulation of the model using a manipulator budget percentage of 5% and a trade rate of 1. As expected, the price begins at approximately $\mu = .30$ before spiking to 1 when the manipulator expends its budget to inflate the market price. After a brief interval, the standard agents purchase the all of the inflated shares, causing the market price to converge back to the equilibrium.

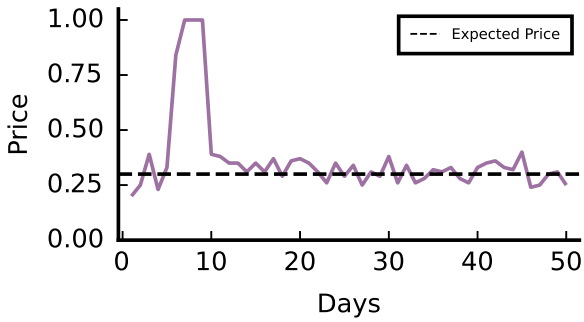


Figure 1: An example model run illustrating a transient manipulation of the market price based on a manipulator who has a budget of 5% of the total budget of the standard agents and a trade rate of 1. Market prices are down sampled to final daily price.

Figure 2 quantifies the magnitude of price manipulation across all combinations of manipulator budget % and trade rate for all prices (top) and final prices (bottom). In the top contour plot, it is clear that both variables have the expected effect: increasing the manipulator's budget % generally leads to stronger price manipulation, whereas increasing the trade rate decreases the effect of price manipulation as it allows standard agents to more quickly restore the equilibrium price.

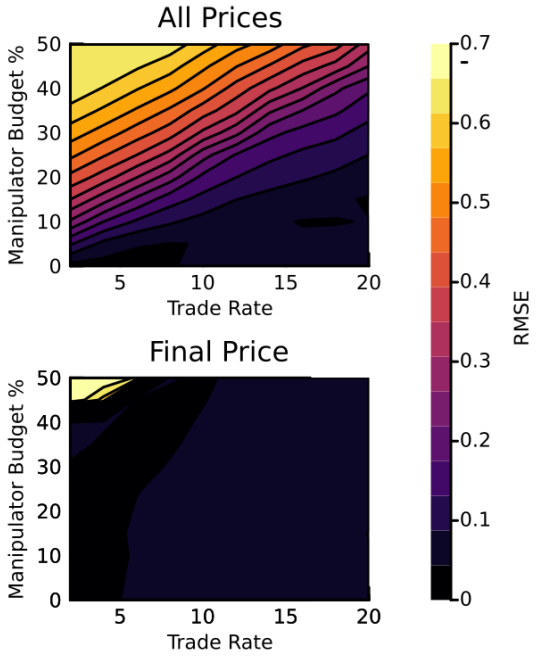


Figure 2: Contour plots showing the effect of the manipulator's budget % and trade rate on all prices and final price, respectively.

The bottom contour plot in Figure 2 shows the effect of manipulator budget % and trade rate on the manipulation of

the final price. Across most of the space, the RMSE is low, indicating that the price manipulation is resolved in most cases by the end of the trading period. However, RMSE is high in the top left quadrant because the standard agents are unable to eliminate the inflated bids within the allotted time (or budget). Within the large space where RMSE is low, there is a small increase in RMSE associated with higher trade rates. We analyzed trade volume dynamics to understand this effect. Our analysis revealed that high levels of trade rate lead to decreased liquidity towards the last part of the trade period, which, in turn, introduced some bias into the market price.

Discussion

Simulation 1 investigated how budget size and trade rate affects the ability of a manipulator to inflate the market price. Our results indicate that a manipulator with a large budget can induce a persistent spike in market price with the duration depending critically upon the trade rate. Assuming the inflated bids are attractive and spur increased trade, one might expect the magnitude of price manipulation to be small to moderate. Under most conditions examined, the market equilibrium was reestablished by the end of the trading period. Nonetheless, price manipulation, even if transient, can diminish the prediction time horizon of a PM. In summary, we found that manipulation can have a wide range of impacts on market price, depending on budget constraints and how quickly the market can respond to a manipulation attempt.

Simulation 2: The Effect of Disinformation

Our goal in Simulation 2 was to investigate the how disinformation can be used as a price manipulation strategy. In this simulation, we investigated the role of two key factors: (1) the degree to which disinformation changes the beliefs of agents (i.e., its persuasiveness), and (2) the interconnectedness of the social network in which the agents are embedded. In general, we expect more effective disinformation messaging will lead to greater distortion of market price, and increasing the interconnectedness of social networks to allow disinformation to spread more quickly and to a greater number of agents, thereby leading to greater price manipulation.

Belief Updating

In simulation 2, agents are identical to the standard agent described in the previous simulation, with two exceptions: (1) each agent can share disinformation with agents connected in its social network, and (2) an agent's beliefs are updated upon initially encountering disinformation. We represent the agent's beliefs in log odds space using a logistic equation. Let $\beta_{k,e} = \log\left(\frac{j_{k,e}}{1-j_{k,e}}\right)$ be agent k's initial belief in log odds, y_k be an indicator variable which assumes a value of 1 if agent k has encountered disinformation, and is 0 otherwise. Let γ represent the increase in log odds that event e will occur upon encountering disinformation. Putting these terms together, the beliefs of agent k log odds is defined as: $\mathrm{LO}_{k,e} = \beta_{k,e} + y_k \cdot \gamma$, which can be transformed back to probabilities via $\frac{1}{1+\exp(-\mathrm{LO})}$.

Network Structure

Agents are embedded in a social network through which disinformation can spread among neighboring (i.e., connected) agents. An important property in terms of disinformation spread is the degree distribution of a social network, which characterizes the extent to which agents are interconnected. We used the Barabási–Albert model to generate a social network, as it approximates many networks found in nature, including power grids, acquaintances among actors, and hyperlinks between web pages (Barabási & Albert, 1999). The Barabási–Albert model generates social networks with a degree distribution that follows a power-law: $P(z) \propto z^{-\alpha}$, where z is the number of neighbors an agent has, and P is the probability mass function (Barabási & Albert, 1999). A power-law distribution is positively skewed such that a few agents have connections to many agents, and many agents have connections only to a few agents (Barabási & Albert, 1999). An important parameter of the Barabási–Albert model is the number of agents added to the network at each step, which we call the growth factor. When the growth factor increases, the social network becomes more densely interconnected, allowing disinformation to spread more easily. The power-law degree distribution is maintained under different values of growth factor; hence, the degree distribution is scale-invariant.

Design Parameters

In this simulation, 500 agents were initialized with a budget of 30 dollars each and submitted orders once per day for a trading period of 50 days. Agents made their decisions sequentially in a different randomized order each day, and the PM operated according to the description in Simulation 1. Each agent's subjective probability was sampled from the same beta distribution used in Simulation 1, and the bid/ask range parameter was set to $\delta = .03$.

On day five, a set of five randomly selected agents were given disinformation, which increased their belief that the event e would occur. After trading on each subsequent day, agents interacted with each of their neighbors one at a time. During each interaction, there was a 5% chance that an agent who had disinformation would share it with its neighbor. If the neighbor received disinformation during the interaction, it could share the disinformation with its neighbors.

We systematically varied the growth factor between 1 and 10 in increments of 1, and the strength of the disinformation γ between 0 and 3 in increments of .25. For each combination of growth factor and disinformation strength, we repeated the simulation 20 times and quantified the effect on market price via RMSE averaged across repetitions as described in Simulation 1.

Results

Figure 3 shows the market price across time for a single run of the simulation. At the beginning, the market price varies around $\mu = .30$. After disinformation is introduced into the social network on day five, it spreads gradually from agent to

agent, increasing the price until the it reaches an asymptote of .75 around day 15.

Figure 4 shows the effect of growth factor and disinformation strength γ on all market prices (top) and the final market price (bottom). In the top contour plot, RMSE for all prices changes as a function of both growth factor and γ. In general, increasing the growth factor increases RMSE because disinformation propagates through the social network more easily. However, the effect of growth factor depends on γ, When γ is low, the effect of growth factor is small. By contrast, when γ is high, growth factor has a negatively accelerated effect on RMSE where the increase is initially fast, but becomes progressively slower with larger values of growth factor.

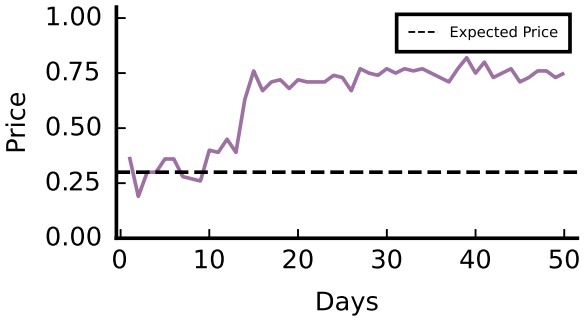


Figure 3: An example model run illustrating the effect of disinformation for disinformation strength $\gamma = 2$ and a growth factor of five. Market prices are down sampled to final daily price.

The contour plot at the bottom of Figure 4 shows the effect of growth factor and disinformation strength γ on final market prices. The results are qualitatively similar to those for all prices, with two notable differences. First, growth factor has a more immediate affect, reaching an asymptote around 2 or 3. Second, RMSE for final prices tends to be larger compared to all prices due to the fact that disinformation has a sustained effect on market prices.

Discussion

Simulation 2 explored the effect of disinformation strength, and the growth factor on price manipulation. Not surprisingly, strength of disinformation had a large effect on manipulating market prices. However, the effect of growth factor was less obvious a priori. Increasing the growth factor had the expected effect of greater price manipulation, but the effect of increasing growth factor diminished quickly around 4 or 5. This suggests that disinformation can spread efficiently in a social network with a power-law degree distribution, likely because there are multiple pathways for disinformation to reach an agent with a large number of neighbors and serve as a hub for disinformation.

General Discussion

Prior empirical research investigating price manipulation in PMs has yielded mixed results, suggesting that successful

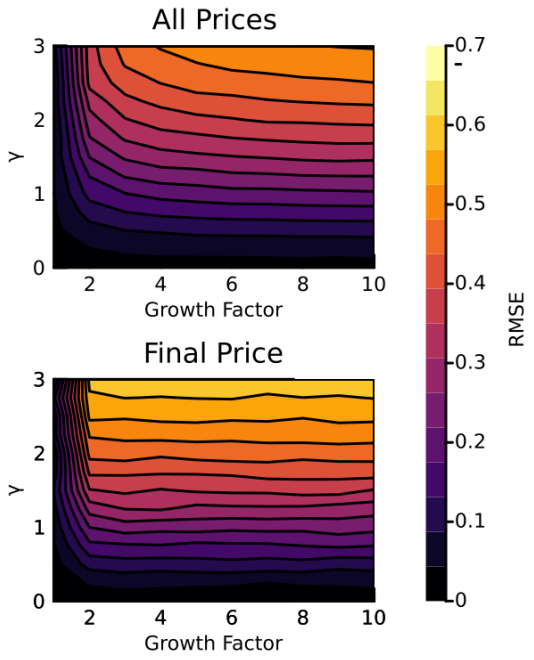


Figure 4: Contour plots showing the effect of growth factor and disinformation strength γ on all prices and final price, respectively.

price manipulation may depend upon poorly understood factors. One approach to understand the sensitivity of PMs to price manipulation is to develop a computational testbed in which a wide range of assumptions regarding manipulation strategy and decision making can be investigated. Using this approach, we investigated the efficacy of two methods of price manipulation across a wide range of conditions using agent-based modeling: (1) a single bad actor who inflates the market price by purchasing a large quantity of shares at a high price, and (2) spreading disinformation in a social network to bias subjective beliefs and consequentially inflate the market price.

Our analysis revealed that a bad actor's attempt to inflate the market price depends critically upon budget constraints and the rate at which other agents can clear the inflated bids from the order book. Even if the inflated bids incentivize a higher trade rate, the magnitude of price manipulation could range from small to moderate. Although an equilibrium was typically reestablished by the end of the trading period, a more persistent problem is that price manipulation can diminish the predictive time horizon of a PM.

Our analysis also revealed that disinformation can spread efficiently through a social network with a degree distribution that follows a power-law. This result depended little on the growth factor, provided that it was at least 4. One reason information spreads efficiently in such networks is that a small set of agents serve as a hub to other agents. As a consequence, there is high chance that a pathway for disinformation exists between any two agents.

Limitations

As with any modeling effort, our results may depend on the assumptions we made. First, in Simulation 1, we assumed (1) standard agents have precise knowledge of their subjective probabilities and (2) the manipulator inundated the order book with a large quantity of bids, resulting in an abrupt and maximal inflation of the market price. Jointly, these two assumptions allow standard agents to identify that the PM is mis-priced and to respond quickly. However, if standard agents instead had a fuzzy or noisy internal representation of subjective probabilities, and the manipulation attempt was more subtle (e.g., increasing the market price from .30 to .40), the price manipulation might be more difficult to detect, and require more time to resolve. Another assumption we made in both simulations, is that subjective probabilities are independent of the market price. Basing subjective probabilities in part on the market price would likely diminish the ability of standard agents to counteract price manipulation.

In Simulation 2, we assumed that disinformation had a static and enduring effect on subjective probabilities. However, the effect of disinformation could be more complex and dynamic due to memory and learning processes: disinformation could be strengthened each time it is encountered, and its strength could decay between encounters. We also assumed a closed system with a fixed number of agents. If additional agents who were insulated from the disinformation entered the PM, they could partially mitigate its effects. Another assumption we made was that agents accepted disinformation without subjecting it to a vetting process. Fact checking could provide a safeguard against disinformation, but this would require that the cost of fact checking is sufficiently low. The cost of fact-checking raises another interesting question: would disinformation spread more easily than factual information given the costs of fact checking and the incentive to keep profitable information private? Further research is needed to address these questions and test the degree to which our results are robust to changes in assumptions.

Conclusion

Although PMs have value as forecasting tools, the conditions under which they can be susceptible to manipulation is not completely understood and is difficult to study. We used agent-based modeling to investigate price manipulation under a variety of conditions. We believe that agent-based modeling could serve as an effective method for stress testing PMs under a wide variety of manipulation strategies and assumptions regarding the decision making and underlying psychology of market participants.

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 Department of the Air Force, or any of their subsidiaries or employees. This research was supported through Air Force internal funds. Distribution A: Approved for public release. Case number: AFRL-2025-0806.

References

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *science*, *286*(5439), 509–512.

Berg, J. E., Nelson, F. D., & Rietz, T. A. (2008). Prediction market accuracy in the long run. *International Journal of Forecasting*, *24*(2), 285–300.

Camerer, C. F. (1998). Can asset markets be manipulated? a field experiment with racetrack betting. *Journal of Political Economy*, *106*(3), 457–482.

Dana, J., Atanasov, P., Tetlock, P., & Mellers, B. (2019). Are markets more accurate than polls? the surprising informational value of "just asking". *Judgment and Decision Making*, *14*(2), 135–147.

Deck, C., Lin, S., & Porter, D. (2013). Affecting policy by manipulating prediction markets: Experimental evidence. *Journal of Economic Behavior & Organization*, *85*, 48–62.

Hansen, J., Schmidt, C., & Strobel, M. (2004). Manipulation in political stock markets–preconditions and evidence. *Applied Economics Letters*, *11*(7), 459–463.

Hanson, R., Oprea, R., & Porter, D. (2006). Information aggregation and manipulation in an experimental market. *Journal of Economic Behavior & Organization*, *60*(4), 449–459.

Hayek, F. (1945). The use of knowledge in society. *American Economic Review*, *35*(4), 519–530.

Johnson, H. M., & Seifert, C. M. (1994). Sources of the continued influence effect: When misinformation in memory affects later inferences. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *20*(6), 1420.

Morton, R. B., Muller, D., Page, L., & Torgler, B. (2015). Exit polls, turnout, and bandwagon voting: Evidence from a natural experiment. *European Economic Review*, *77*, 65–81.

Page, L., & Clemen, R. T. (2013). Do prediction markets produce well-calibrated probability forecasts? *The Economic Journal*, *123*(568), 491–513.

Wolfers, J., & Zitzewitz, E. (2004). Prediction markets. *Journal of Economic Perspectives*, *18*(2), 107–126.

Wolfers, J., & Zitzewitz, E. (2006). *Interpreting prediction market prices as probabilities.* National Bureau of Economic Research Cambridge, Mass., USA.

Exploring Integrated Co-occurrence and Semantic Mechanisms for Long Term Memory Retrieval

Lily Gebhart (gebhart@oxy.edu)
Justin Li (justinnhli@oxy.edu)
Occidental College, 1600 Campus Road
Los Angeles, CA 90041, USA

Abstract

Semantic and co-occurrence memory associations aid the retrieval of relevant memory elements from long term memory, but little is understood about how semantics and co-occurrence interact in this process. This paper explores the relationship between these associations via computational memory modeling in a Bayesian framework. We assessed the performance of eleven candidate mechanisms on two linguistic tasks - the Word Sense Disambiguation task and the Remote Associates Test. The most successful mechanisms use co-occurrence associations to modulate semantic associations by removing from or adding to the context or pool of candidates for retrieval, consistent with recent experimental work in memory retrieval. Although these results are a promising first step for understanding the relationship between semantic and co-occurrence associations in memory retrieval, more empirical human data is needed to validate the proposed interactions between these associations.

Keywords: Long-term Memory Retrieval; Semantics; Co-occurrence; Bayesian Memory

1 Introduction

Memory retrieval is a complex process, dependent on factors including past retrieval history, working memory context, and associations between elements in long-term memory. These associations could come from many sources; two common sources are that of co-occurrence and semantic associations, which have both been shown to affect memory retrieval. The effects of semantic and co-occurrence associations have often been studied separately, both experimentally and with computational modeling (Schatz, Jones, & Laird, 2022; Greenberg & Verfaellie, 2010). Recent experimental work in psychology and neuroscience, however, suggests that the relationship between co-occurrence and semantics is also important to their role in memory retrieval, though the nature of this relationship has yet to be fully explored (Manning, Sperling, Sharan, Rosenberg, & Kahana, 2012; Ferreira, Charest, & Wimber, 2019; Greenberg & Verfaellie, 2010). For cognitive modeling in particular, previous computational memory models have not consistently defined co-occurrence and semantic retrieval mechanisms, or considered their relationship to be important to their functionality, despite recent experimental progress (Chater et al., 2020; Roelke et al., 2018; Hofmann, Kleemann, Roelke-Wellmann, Vorstius, & Radach, 2022).

In this paper, we investigate the nature of the relationship between semantics and co-occurrence using Bayesian memory models. Using computational models, rather than experimental approaches, has the potential to clarify the relationship between semantics and co-occurrence for human long term memory retrieval and representations of this process in cognitive models. Computational models enable the exploration of possible memory retrieval mechanisms in a simplified framework that can be compared to experimental data later on if necessary. As such, with this work, we explore the space of potential relationships between semantics and co-occurrence for long term memory retrieval with the aim of illuminating psychological features of their relationship and inspiring its consideration in future computational modeling implementations and psychological experiments. [1]

2 Bayesian Memory Modeling

Co-occurrence and semantics refer to associations between memory elements that aid and influence long term memory retrieval. As an example, consider the words “animal”, “rat”, and “pack”. The words “rat” and “animal” have a semantic association since rats are animals; on the other hand, “rat” and “pack” have a co-occurrence association, since they are more likely to occur together in the compound word, “packrat”. In this paper, we are interested in how the presence of both “animal” and “pack” together might influence the retrieval of “rat”.

We explore this question within a probabilistic Bayesian approach, commonly used in psychological models to determine which memory elements to retrieve into working memory given some immediate context (Tulving & Craik, 2005). Evidence suggests that human memory is only approximately Bayesian, since it is subject to error in encoding, storage, and recall, and is limited to only the biased group of experiences we’ve encountered (Chater et al., 2020). Nonetheless, in the idealized case, the memory retrieved m is one from the set of viable retrieval candidates M with the highest posterior probability given some working memory context C; that is,

$$\underset{m \in M}{\operatorname{argmax}}\ P(m|C) = \underset{m \in M}{\operatorname{argmax}}\ \frac{P(C|m)P(m)}{P(C)}$$

$P(m)$ and $P(C)$ do not depend on the relationship between co-occurrence and semantics, and thus we are primarily interested in the likelihood $P(C|m)$, which is defined differ-

[1] The code for this paper is available at https://github.com/Lily-Gebhart/Exploring-Integrated-Mechanisms

ently for retrieval mechanisms such as semantics and co-occurrence.

For semantic retrieval, $P(C|m)$ is calculated based on spreading activation and base-level activation, which has been shown to reflect a Bayesian estimate of need odds (Anderson & Milson, 1989; Anderson & Schooler, 1991). Our SEMANTIC retrieval agent implements this mechanism over task-specific semantic networks. When a memory element is retrieved, it is *activated*, with additional diminished activation spreading throughout the connections of a semantic network, beginning with neighbors of the retrieved memory element (Thomson & Lebiere, 2013; Taatgen, Lebiere, & Anderson, 2006). Spreading activation is defined as

$$P(C|m) \propto \text{act}_m = \sum_{k=1}^{n} \frac{1}{2^{\text{dist}(m,k)}} t_k^{-d}$$

where the activation for the memory element of interest m is the sum of the spreading activation it has received previously from each of the n memory elements k, with $\text{dist}(m,k)$ the distance between memory elements m and k in the semantic network, t_k the time since the spreading activation from k occurred, and d a time decay term.

For co-occurrence, $P(C|m)$ is extracted from the statistical co-occurrence of words or concepts in natural language, approximated through sources such as the Google Books ngrams database (Michel et al., 2011) and SemCor corpus (Miller, Leacock, Tengi, & Bunker, 1993). We make the simplifying assumption that the co-occurrence probabilities between each candidate memory element m_i and each context memory element c_j in the current working memory context C are independent, meaning that the likelihood is the product of all $P(m_i|c_j)$ probabilities. This mechanism is the basis of our CO-OCCURRENCE agent that uses task-specific statistics.

3 Integrated Mechanisms

While the likelihood term is well defined for semantics and co-occurrence associations separately, we are interested in the question of how it might be defined when both associations are involved. In Table 1 below, we define eleven candidate mechanisms that explore the relationships between the semantic and co-occurrence associations. These mechanisms can be organized into four categories based on the nature of the psychological relationships they capture, with the goal of exploring the space of potential psychological relationships between these associations.

Two of the mechanisms assume that semantics and co-occurrence are *Independent* in the memory retrieval process, and combine their respective probabilities accordingly. Since semantics and co-occurrence have been primarily studied in isolation from each other in the literature (Greenberg & Verfaellie, 2010), our proposed mechanisms make the simplifying assumption that they are derived from different sources with separate roles in memory retrieval. Two other mechanisms also consider semantic and co-occurrence information as probabilities, but determine retrieval candidates based on features of the semantic and co-occurrence probability distributions, namely their variance and maxima. Intuitively, higher confidence retrievals will result when the retrieved candidates have significantly higher retrieval probabilities than the other viable candidates; these *Distribution-Based* mechanisms will then select the result from the mechanism that has more distinct, or more confident, retrieved candidates.

The remaining categories of mechanisms consider how semantic associations might modify co-occurrence retrieval mechanisms or vice versa. Each of these modification affects different parts of the Bayesian likelihood probability $P(C|m)$ introduced above. The *Probability Modification* category mechanisms use co-occurrence or semantics to directly modify the retrieval probabilities of semantics or co-occurrence, respectively. These mechanisms suggest that co-occurrence and semantic relationships are highly intertwined, and that the calculation of likelihood depends on both associations at a deep level. The *Context Modification* mechanisms add additional memory elements to the context for semantics or co-occurrence, and the *Candidate Modification* mechanisms add or remove memory elements from the pool of viable retrieval candidates. This affects the context C and the retrieval candidates M respectively, since altering the number of memory elements in the context or the number of elements considered as viable retrieval candidates will affect the result of the overall memory retrieval. The intuition behind these mechanisms is to use one association to either narrow the focus of, or to fill in missing information from, the other association.

4 Task Descriptions

We evaluate these candidate co-occurrence and semantic relationships with two computational tasks: Word Sense Disambiguation (WSD) and the Remote Associates Test (RAT). Both tasks have been used for assessing long term memory retrieval, especially in computational modeling applications (Kwong, 2012; Dutta & Basu, 2012; Schatz et al., 2022; Marko, Michalko, & Riečanskỳ, 2019). We selected these tasks for exploring and evaluating the different integrated mechanisms because it has been demonstrated that co-occurrence retrieval mechanisms outperform semantic mechanisms on the WSD task (Montoyo, Suárez, Rigau, & Palomar, 2005; Krovetz & Croft, 1992) and semantic mechanisms outperform co-occurrence mechanisms on the RAT (Schatz et al., 2022). This tradeoff enables us to explore how different candidate integrations of semantics and co-occurrence perform on co-occurrence and semantics-based retrieval tasks. Here, we review the rationale for this tradeoff and describe how we implemented the standard CO-OCCURRENCE and SEMANTICS agents for each task.

4.1 Word Sense Disambiguation (WSD) Task

The WSD task asks an agent to identify the sense of a word given other context words in the sentence. For example, "pack" in the sentence "The pack of rats were swimming" could mean either a group of animals or a small container;

Category	Mechanism	Description
Independent	Joint Probability (JPR)	The probability distributions of SEMANTICS and CO-OCCURRENCE are assumed to be mathematically independent, and the product of probabilities corresponding probability elements is taken to produce a joint probability distribution. This suggests that co-occurrence and semantics are derived from independent sources and contribute to the retrieval process without additional interactions.
	Additive Probability (APR)	The probability distributions of the SEMANTICS and CO-OCCURRENCE agents for each trial are added together. This is proportional to the average of the two distributions. Similar to the previous mechanism, this suggests that the distributions have minimal interaction before they are involved in memory retrieval and give separate estimates of the probability that is averaged over.
Distribution Based	Maximum Probability (MPR)	One of the results from the standard SEMANTICS or CO-OCCURRENCE mechanisms is used, depending on which has a higher probability. Here, co-occurrence and semantics are not directly combined, but one mechanism or the other is selected on a per-retrieval basis, based on which has less uncertainty, as estimated by the maximal probability memory element(s).
	Variance-Based Selector (VBS)	One of the results from the standard SEMANTICS or CO-OCCURRENCE mechanisms is used, depending on which distribution has higher "certainty". We explored two definitions of certainty: as the standard deviation of probabilities of the candidates, and as the difference between the highest and second highest maximal probability memory element(s) in each distribution.
Probability Modification	Semantics Boosted Co-occurrence (SBC)	If two words are semantically related, their co-occurrence conditional probabilities will receive a small boost according to some function $f : [0,1] \rightarrow [0,1]$. We explored two such functions, the sigmoid and square root, for the semantic "boost". This implies that semantic associations in memory retrieval further boost existing co-occurrence associations.
	Co-occurrence Weighted Semantics (CWS)	The edges between memory elements in the semantic network are weighted by conditional co-occurrence probabilities. This implies that co-occurrence associations in memory retrieval is modulate the degree to which memory elements are semantically related in the network.
Context Modification	Semantic Supplemented Co-occurrence (SSC)	Memory elements that are directly semantically related to each of the original context memory elements are also included as context; with this expanded context, the standard co-occurrence retrieval mechanism is used. This suggests that semantic associations are used to supplement the current working memory context for co-occurrence memory retrieval.
	Co-occurrence Supplemented Semantics (CSS)	Memory elements that are co-occurrence associations to the original semantic context are also included as context; this expanded context is activated as part of the standard semantic retrieval mechanism. Similar to SSC, this suggests that co-occurrence associations are used to supplement the current working memory context for semantic memory retrieval.
Candidate Modification	Co-occurrence Expanded Semantics (CES)	All co-occurrence associations are incorporated into the semantic network as if they were semantic associations themselves. This allows spreading to occur over co-occurrence associations as well, with the implication that co-occurrence associations help define the semantic network structure.
	Co-occurrence Filtered Semantics (CFS)	Associations between memory elements in the semantic network that do not also have a co-occurrence association are removed, before the standard semantic mechanism is used. This suggests that stored co-occurrence associations filter the semantic network so that only the most relevant semantic associations remain.
	Semantic Filtered Co-occurrence (SFC)	Co-occurrence associations that are not also semantically associated are removed, before the standard co-occurrence mechanism is used. This implies that co-occurrence associations between two memory elements are only maintained if the semantic activation probability between the elements is above a predefined threshold.

Table 1: Integrated Mechanisms and Agents

deciphering its meaning is crucial to understanding what the speaker or writer means. Past literature has shown that the WSD task relies heavily on statistical co-occurrence information (Montoyo et al., 2005), making it a meaningful metric for how candidate integrated mechanisms are able to utilize co-occurrence information in memory retrieval.

Our implementation of the WSD task uses text from the SemCor corpus. A subset of WordNet that contain the semantic relations of words in the SemCor corpus were used to generate the semantic network (Bird, Klein, & Loper, 2009). We further divided the corpus into six partitions of 5,000 sentences to limit the spreading of activation and make our models more computationally tractable. We measure task performance as the percentage of correct trials averaged over all partitions. If the agent suggests that n senses are equally likely and one of them is correct, the agent's performance is discounted proportionally as $\frac{1}{n}$.

In each trial, the SEMANTICS agent activates all other words in the sentence at the same time. Spreading activation then occurs, and the most activated element is used as the answer. The decay parameter and the spreading activation depth were kept constant. The SEMANTICS WSD agent is additionally parameterized by whether how far activation spreads, and by whether the network activations are cleared after every trial, cleared after all trials in a sentence, or never cleared; otherwise, the repeated activation of the words in each sentence accumulates.

Meanwhile, the CO-OCCURRENCE agent answers using word with the maximum posterior probability, with a naive Bayes assumption for all context words. The probabilities

are learned from the SemCor corpus itself, but we parameterize the agent by either considering the correct senses of the context words, or only the context words themselves. For example, in the sentence, "The pack of rats were swimming", the conditional probability for the retrieved sense of "pack" could be based on co-occurrence with the correct sense of "rats", or merely the co-occurrence with the word "rats" regardless of its sense.

4.2 Remote Associates Test (RAT)

In RAT, an agent is provided with three context words and asked to find a word that relates to all three. For example, if given the words "animal", "back", and "rat", the correct response would be "pack" to form the compound words "pack animal", "backpack", and "pack rat" (Bowden & Jung-Beeman, 2003). RAT is often used as a test of creativity and mental acuity in experimental psychology, with relatively poor human performance due to the limited context given in the task (Wu, Huang, Chen, & Chen, 2020). As such, it is also a good test of whether candidate integrated mechanisms are able to utilize relational, semantic associations, although many of the solutions to RAT problems also rely on commonly co-occurring words.

Our implementation of the RAT uses the 142 question bank from Bowden and Jung-Beeman (2003), which is frequently used in both experimental and computational studies. Given the three context words for each trial, each agent only guesses once, selecting the candidate target with the maximum probability or activation as its guess (Schatz et al., 2022). RAT is a difficult task in general, with performance here further constrained by the fact that the co-occurrence and semantic data sources do not contain solutions to every RAT problem. Therefore, it is reasonable to expect low accuracy on the RAT in general, regardless of the mechanism.

The SEMANTICS agent uses a semantic network generated from the South Florida Free Association Norms (SFFAN) corpus. Since the SFFAN corpus is a large database of human free association norms, it may contain some co-occurrence relations in addition to semantic relations (Nelson, McEvoy, & Schreiber, 2004). While prior work have used other semantic networks (Schatz et al., 2022), SFFAN contains many of the RAT associations in a small network, and therefore is relatively computationally tractable for use in experiments. In each trial, the agent activates all context words, then uses the word with the highest resulting activation (that is not also a context word) as the answer. All activations in the semantic network were cleared after each trial.

The CO-OCCURRENCE agent uses co-occurrence conditional probabilities from the Google Books English One Million dataset, which provides a relatively unbiased, representative source of co-occurrence probabilities required for the RAT (Michel et al., 2011). We only use bigrams and not all n-grams for computational tractability and consistency with the smaller network sizes used in our implementation of the WSD task. In each trial, for each candidate answer, its bigram probabilities with all three context words is multiplied together; the word with the highest joint probability is selected as the trial guess. The task accuracy is computed using the same methods employed to compute the accuracy on the WSD.

4.3 Task Baselines

To contextualize the standard SEMANTICS and CO-OCCURRENCE agents, as well as the integrated agents' performance on the WSD task and RAT, we compare their performance to two additional baselines: the ORACLE and uniform RANDOM agents. The ORACLE is correct if either of CO-OCCURRENCE or SEMANTICS is correct, thereby indicating the expected maximum performance considering only the abilities of the SEMANTICS and CO-OCCURRENCE agents. It is possible for an integrated agent to perform better than ORACLE if it is able to find synergy between co-occurrence and semantic information. The ORACLE is therefore not necessarily an upper bound on the performance of integrated agents.

The uniform RANDOM baseline r assesses how well integrated candidates perform relative to chance. For each of n trials, $r = \sum_{i=1}^{n} \frac{1}{s_i}$. For the WSD task, s_i is the number of valid word senses of each trial and for the RAT task, s_i is the number of words that form bigrams with all three trial words; This s_i value for the RAT task provides a better comparison of performance than statically using the size of the SFFAN network because many of the candidates in the network have zero probabilities each trial and the candidates with non-zero probabilities fluctuate between trials.

5 Experiment Results

Figure 1 provides a comparison of the relative performance of each agent on the WSD and RAT tasks. For simplicity, for each agent, we only show the results of the most successful combination(s) of agent and task parameters. As expected, there is a tradeoff in the performance of the standard CO-OCCURRENCE and SEMANTICS agents, with CO-OCCURRENCE outperforming the SEMANTICS on WSD and vice versa on RAT, confirming our initial assumptions. These results and the results of the integrated agents to follow are further contextualized by the performance of the ORACLE (WSD: 100%, RAT: 37.3%) and RANDOM baselines (WSD: 37.9%, RAT: 3.1%), representing the maximum anticipated and chance-level performance on each task.

In general, the results of the integrated agents on either task are not consistent with the original psychologically-based framework for organizing the integrated mechanisms. For each mechanism the retrieval probabilities of all viable retrieval candidates for each trial results in a probability distribution; the average variance and ranking of the solution in each trial distribution for a given agent are used to interpret the results of each agent on either task. It is relevant to note that there is a much larger number of candidates with nonzero probabilities for the RAT compared to the WSD, which contributes to the differences in distribution shape on each task as discussed below. Additionally, mechanisms perform better

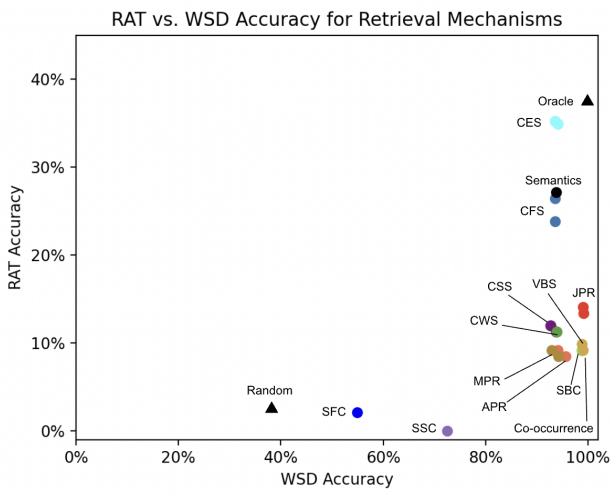


Figure 1: A performance comparison of the accuracies of the Co-occurrence, Semantic Spreading, and integrated mechanism candidates on WSD and RAT.

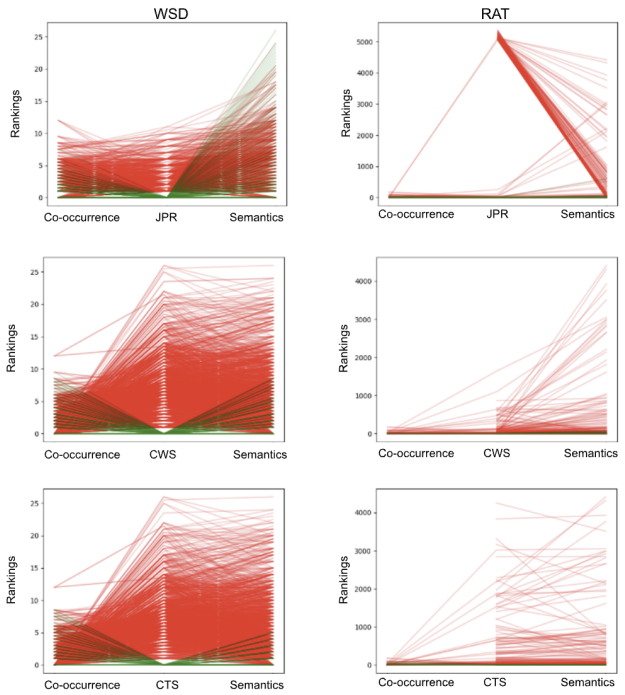


Figure 2: The rankings of the solution to each task trial in CO-OCCURRENCE, SEMANTICS, and integrated agent distributions is compared for three integrated agents: CWS, JPR, and CTS. Lines connect the ranking of the solution in each of the CO-OCCURRENCE, integrated, and SEMANTICS distributions for a given trial with green lines indicating that the integrated agent correctly retrieved the solution element for the task and red lines that it did not.

on WSD when the variance of the distribution of their candidate retrieval probabilities is higher. Higher variance implies that distributions are more "peaked", or that the highest probability retrieval candidates are significantly more likely to be the retrieved than the other candidate elements. The same trend does not hold for RAT.

We will now discuss how the results of each integrated agent contributes to our understanding of the relationship between semantics and co-occurrence in facilitating long term memory retrieval.

The SFC and SSC agents perform poorly on both the WSD and RAT, indicating that integrating semantic information into the context or candidate pool of a co-occurrence based retrieval is ineffective for retrievals that largely rely on co-occurrence (WSD) or semantic (RAT) information. The poor performance of SFC can be attributed to the fact that influential co-occurrence associations that are not also semantic associations fail to be considered as retrieval candidates. Meanwhile, the SSC agent performs poorly on the WSD and RAT likely as a result of reduced variance between candidates, or equivalently, increased ambiguity of which candidate is correct; increasing the number of memory elements in the context results in a more diffuse retrieval probability distribution, lessening the likelihood that the ideal candidates will be retrieved.

The CSS, CWS, MPR, and APR agents comprise a cluster of mechanisms which perform about the same as CO-OCCURRENCE on RAT and about the same as SEMANTICS on WSD. The success of these agents can be attributed to factors specific to each mechanism including high retrieval ambiguity for APR, the ineffectiveness of selecting the maximum probability element as a proxy for "certainty", and increased selectivity by weighting and removing semantic network edges by CWS and CSS. Interestingly, though CWS performed more poorly than SEMANTICS on RAT, it does increase the ranking of the correct retrieval candidate compared to the CO-OCCURRENCE and SEMANTIC agents (shown in Figure 2. It is possible that more complex integrations of co-occurrence retrieval probabilities into determining edge weights are necessary for more effective retrieval using this strategy, but such weights were not explored here.

Another cluster of candidates including JPR, SBC, and VBS performs nearly identically to CO-OCCURRENCE on WSD and CO-OCCURRENCE on RAT, with JPR performing slightly better than the other candidates on RAT. As opposed to the other independent mechanism, APR, which suffers from high retrieval ambiguity, JPR seems promising as a candidate retrieval mechanism. As demonstrated prominently in the RAT ranking comparison plot in Figure 2, JPR generally ranks the correct retrieval candidate higher in the distribution, as long as the solution is both co-occurrently and semantically related to the context, otherwise it is ranked low in the distribution. Testing the JPR on different sources of co-occurrence and semantic associations may be useful in gathering more conclusive results on the efficacy of JPR, especially for semantically-based retrievals.

Finally, the CFS and CES agents perform best out of all integrated candidates, with performance similar to SEMANTICS on WSD and similar or better than SEMANTICS on RAT. The performance of CFS suggests that the removal of non co-occurrence associations from the network maintains

edges that link context words to the ideal retrieval candidates while removing less relevant connections from the network. However, the random effects the removal of these candidates has on the ranking of the solution in each trial suggests that critical edges are also being removed from the network, as shown in Figure 2. Removing edges from the graph based on the strength rather than the lack of existence of co-occurrence associations as in the current CFS may prove more effective in both co-occurrence and semantic based retrievals. Meanwhile, the exceptional performance of CES, especially on RAT, may be partially attributed to the inclusion of co-occurrence relations in the free-association corpora used to generate the semantic network. Furthermore, by nature of the corpora used, each word has more co-occurrence associations than semantic associations, so the effect may have resulted from heightened spreading activation in the larger more interconnected network that resulted. Regardless, these results suggest that co-occurrence associations are helpful in modifying the structure of the semantic network for both co-occurrence and semantic based retrieval tasks. Further investigation is necessary to determine if adding and deleting edges from the semantic network based on co-occurrence associations would be more effective than CFS and CES.

6 Discussion

The computational agents implemented in this paper, and the underlying retrieval mechanisms they represent, explored several possible relationships between semantic and co-occurrence mechanisms in long term memory retrieval. Based on our results, the most successful agent across the two tasks uses co-occurrence associations to modulate the semantic retrieval mechanism by adding relevant co-occurrence associations to the retrieval context, which broadens the search over the semantic network. Other successful agents on both tasks modify the semantic network by adding or removing edges based on co-occurrence associations. In both cases, the core underlying retrieval mechanism relies on semantic networks, but co-occurrence associations provide additional information that modify the spreading activation process. While each of these co-occurrence influences on semantic retrieval were explored in independent mechanisms, it is likely that some or all of these mechanisms are at play, with co-occurrence associations modulating semantics at multiple stages in the retrieval process.

In a way, these results fit in with findings from psychology literature. Children learn co-occurrence associations at a young age by first associating concepts that directly co-occur in their natural environment, and only as their linguistic and conceptual abilities mature do children learn semantic associations as memory elements with similar sets of co-occurrence associations (Unger & Fisher, 2021; Savic, Unger, & Sloutsky, 2023). Recent developments in word embedding models such as word2vec further demonstrate how semantic relationships could be derived from natural usage statistics (Rohde, Gonnerman, & Plaut, 2006). In both children and such AI models, semantic associations serve as symbolic gists of the underlying co-occurrence statistics, but may not capture the nuances of the full joint probabilities. While many tasks may be solvable using semantic associations, it is possible that additionally incorporating lower-level co-occurrence information - using it to introduce additional context or to (de)emphasize particular semantic associations - could lead to better retrievals. If we consider episodic memory to be a source of co-occurrence information, this may also corroborate accounts that episodic memory facilitates retrieval from semantic memory (Greenberg & Verfaellie, 2010).

Even though our computational results are suggestive of trends in experimental work, it is important to note that this remains only an early step in understanding the relationship between co-occurrence and semantic associations in memory retrieval. While the mechanisms suggested several ways that semantics and co-occurrence may interact, there are many other possible mechanisms for their interaction. We did not explore mechanisms that integrate the associations in more complex ways, nor how multiple mechanisms could be incorporated into a larger framework. Furthermore, while we considered several variable parameters for the tasks and agents considered, there are a large number of parameters that we did not consider or left fixed in our models. It is also likely the case that co-occurrence associations are only independent in certain cases, not in general as we assumed here.

One difficulty is that co-occurrence and semantic effects have been surprisingly difficult to tease apart experimentally; we were only able to find one paper that attempts to measure how these associations affect retrieval separately and together (Roelke et al., 2018). This dearth of human data make modeling difficult, hence our use of simple linguistic tasks, which bring their own problems regarding the datasets and database from which we draw co-occurrence and semantic information. Beyond how these associations may not match those in people, the distinction between semantic and co-occurrence is also blurred in the data, as the semantic networks used in our experiments also contain co-occurrence associations. As mentioned above, such gray areas between different types of associations also exist in people, which further complicates efforts understand the distinct and combined contributions.

Nonetheless, although this work is preliminary, we believe it is a needed first step to combine the literature on co-occurrence and semantic associations. Computational models have not cleanly distinguished between the effects of these mechanisms, and have even used the same representations (such as semantic networks) to model both leading to difficulties in understanding their separate and joint effects on memory retrieval. The psychology literature has been much more careful in considering each association, and to follow suit, we carefully considered how they might be different and how their effects might be combined in a computational model. Much additional work is needed, both experimentally and via modeling, to fully understand the interplay of co-occurrence and semantic associations in memory retrieval.

References

Anderson, J. R., & Milson, R. (1989). Human memory: An adaptive perspective. *Psychological Review*, *96*(4), 703.

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

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly.

Bowden, E. M., & Jung-Beeman, M. (2003). Normative data for 144 compound remote associate problems. *Behavior research methods, instruments, & computers*, *35*, 634–639.

Chater, N., Zhu, J.-Q., Spicer, J., Sundh, J., León-Villagrá, P., & Sanborn, A. (2020). Probabilistic biases meet the Bayesian brain. *Current Directions in Psychological Science*, *29*(5), 506–512. doi: 10.1177/0963721420954801

Dutta, S., & Basu, A. (2012). A cognitive approach to word sense disambiguation. In *Proceedings of the 13th international conference on computational linguistics and intelligent text processing* (pp. 211–224).

Ferreira, C. S., Charest, I., & Wimber, M. (2019). Retrieval aids the creation of a generalised memory trace and strengthens episode-unique information. *NeuroImage*, *201*, 115996.

Greenberg, D. L., & Verfaellie, M. (2010). Interdependence of episodic and semantic memory: Evidence from neuropsychology. *Journal of the International Neuropsychological society*, *16*(5), 748–753.

Hofmann, M. J., Kleemann, M. A., Roelke-Wellmann, A., Vorstius, C., & Radach, R. (2022). Semantic feature activation takes time: longer soa elicits earlier priming effects during reading. *Cognitive Processing*, *23*(2), 309–318.

Krovetz, R., & Croft, W. B. (1992). Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems (TOIS)*, *10*(2), 115–141.

Kwong, O. Y. (2012). *New perspectives on computational and cognitive strategies for word sense disambiguation*. Springer Science & Business Media.

Manning, J. R., Sperling, M. R., Sharan, A., Rosenberg, E. A., & Kahana, M. J. (2012). Spontaneously reactivated patterns in frontal and temporal lobe predict semantic clustering during memory search. *Journal of Neuroscience*, *32*(26), 8871–8878.

Marko, M., Michalko, D., & Riečanskỳ, I. (2019). Remote associates test: An empirical proof of concept. *Behavior research methods*, *51*(6), 2700–2711.

Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Team, G. B., ... others (2011). Quantitative analysis of culture using millions of digitized books. *Science*, *331*(6014), 176–182.

Miller, G. A., Leacock, C., Tengi, R., & Bunker, R. T. (1993). A semantic concordance. In *Human language technology: Proceedings of a workshop held at plainsboro, new jersey, march 21-24, 1993.*

Montoyo, A., Suárez, A., Rigau, G., & Palomar, M. (2005). Combining knowledge-and corpus-based word-sense-disambiguation methods. *Journal of Artificial Intelligence Research*, *23*, 299–330.

Nelson, D. L., McEvoy, C. L., & Schreiber, T. A. (2004). The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, *36*(3), 402–407.

Roelke, A., Franke, N., Biemann, C., Radach, R., Jacobs, A. M., & Hofmann, M. J. (2018). A novel co-occurrence-based approach to predict pure associative and semantic priming. *Psychonomic Bulletin and Review*, *25*(4), 1488–1493. doi: 10.3758/s13423-018-1453-6

Rohde, D. L., Gonnerman, L. M., & Plaut, D. C. (2006). An improved model of semantic similarity based on lexical co-occurrence. *Communications of the ACM*, *8*(627-633), 116.

Savic, O., Unger, L., & Sloutsky, V. M. (2023). Experience and maturation: The contribution of co-occurrence regularities in language to the development of semantic organization. *Child development*, *94*(1), 142–158.

Schatz, J., Jones, S. J., & Laird, J. E. (2022). Modeling the remote associates test as retrievals from semantic memory. *Cognitive Science*, *46*(6), e13145.

Taatgen, N. A., Lebiere, C., & Anderson, J. R. (2006). Modeling paradigms in ACT-R. *Cognition and multi-agent interaction: From cognitive modeling to social simulation*, 29–52.

Thomson, R., & Lebiere, C. (2013). Constraining Bayesian inference with cognitive architectures: An updated associative learning mechanism in ACT-R. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 35).

Tulving, E., & Craik, F. I. (2005). *The oxford handbook of memory*. Oxford University Press.

Unger, L., & Fisher, A. V. (2021). The emergence of richly organized semantic knowledge from simple statistics: A synthetic review. *Developmental Review*, *60*, 100949.

Wu, C.-L., Huang, S.-Y., Chen, P.-Z., & Chen, H.-C. (2020). A systematic review of creativity-related studies applying the remote associates test from 2000 to 2019. *Frontiers in psychology*, *11*, 573432.

Decisions in Tetris are Often Hard: Mapping the Difficulty of Decision Points in a Complex, Dynamic Task

Theodros M. Haile (T.M.Haile@rug.nl)

Catherine Sibert (C.L.Sibert@rug.nl)
Bernoulli Institute of Computer Science, Mathematics and Artificial Intelligence
University of Groningen,
The Netherlands

Abstract

AI tools offer the promise of individualized training and support in a variety of tasks, but to determine when and what kind of assistance to provide, it will be necessary for such tools to gauge the level of difficulty of each action within the overall task environment. This becomes more challenging in dynamic environments, where many small-scale decisions or actions contribute to a long term goal, but no individual action can be objectively labeled as being "correct". Here, we attempt to map the difficulty of just such a task: the video game Tetris. Using a model that is capable of human-like expert performance, we take advantage of the model's ability to evaluate and rate the optimality of all possible actions at every decision point. Decisions with a single action that is rated definitively higher than any other option are considered easy, and and decisions with multiple plausible options are rated as more difficult in proportion to the number of reasonable actions. We look at the incidence rate of easy and difficult decisions, and how it varies across players of different skill levels.

Keywords: Decision making, Difficulty, Cognitive Modeling, Expertise

Introduction

As artificial intelligence technology continues to improve and develop, it is becoming increasingly possible to create individualized tools to optimize learning and support high-level performance in a variety of task domains. But while AI systems have demonstrated much promise in natural language processing, and are being tested widely in simple task completion (Shen et al., 2023), education (Kasneci et al., 2023) and even medicine (Liévin, Hother, Motzfeldt, & Winther, 2024), it has been more challenging to apply them in complex and dynamic task environments, particularly those where any individual action is a small part of a sequence that eventually produces some kind of measurable outcome. For tasks like this, identifying when and which type of action is challenging for an individual could provide valuable insight into when and what type of assistance would be most helpful. Even more so, if this assistance could be offered in real time. In this project, we propose to use an AI model to gauge the relative difficulty of individual actions within a complex, dynamic decision making task. Such classification could then be compared against behavioral data of humans performing the same task, to determine if the model is capable of reliably identifying specific challenging points for individuals within the larger context of the task. This would provide a first proof-of-concept towards building systems capable of providing real-time decision support in the learning and execution of other complex dynamic tasks.

Decision Making in Dynamic Tasks

The outcome of any decision, even in simple tasks, depends on many factors. Some of these factors relate to the task and its environment, such as the characteristics of the decision space or the number of perceived outcomes. Others depend on the decision maker, like cognitive traits and prior knowledge. The interaction between these factors determines, among other things, how difficult an individual decision-maker will find a given choice within a task. In this project, we focus on dynamic tasks, and propose a method to gauge decision difficulty.

Most decision-making research involve simple, static laboratory tasks that change one or two dimensions at a time. These involve learning categories or navigating decision trees, where numbers of stimulus features (Ashby & Maddox, 2011), complexity (A. G. Collins, 2018; Gläscher, Daw, Dayan, & O'Doherty, 2010), or reliability (low state-transition probabilities (A. Collins & Koechlin, 2012)) are altered. In such cases, it is fairly straightforward to teach an individual an optimal strategy to complete the task, and simple computational or AI models are able to perform them quite well (A. G. Collins, 2018). The vast majority of real decisions, however, are made in naturalistic, dynamic conditions, and strategies must adjust as conditions change. This means that task features and decision points change within the environment, both as a result of the system's intrinsic rules and from the sequence of previous decisions and actions that a user has made (Brehmer, 1992; Gonzalez, Lerch, & Lebiere, 2003). A dynamic environment significantly increases the complexity of any effort to model a decision space, and often adds the feature that taking no action, a relatively neutral action in a static space, becomes a more active decision as the environment can change without the user's intervention.

The effects of the environment and the user's actions on the decision space are not independent, and forming a model of the interaction that can track changes and guide future actions is necessary for any learner to perform the task with skill. But individuals' information processing abilities and prior experience vary widely, adding to the challenge. Domain expertise can be thought of as a set of intuitions that allow a user to quickly recognize large numbers of features

that reveal relevant patterns in the environment which guide the efficient selection of beneficial actions (Hutton & Klein, 1999). Novices, lacking these intuitions, may not be aware of particular features, focus on irrelevant patterns, or incorrectly associate irrelevant features to desired outcomes, leading to less efficient, suboptimal choices ((Hutton & Klein, 1999); (Shanteau, 1988). Successful mapping of features to outcomes is vital to skill acquisition, but as tasks become more complex, the distance between an individual action and a tangible benefit grows and it is not always obvious which possible action is truly optimal at any given decision point. When these decisions are placed within the context of a dynamic environment, even if a utility calculation of all available choices is technically possible, it is often not worth the cost when an approximation rule could be applied more quickly. As such, it is often challenging to evaluate performance at a decision level, and success is instead measured with longer-term outcome measures. But as every action contributes to that outcome, the ability to evaluate the quality of individual actions within dynamic environments would be enormously beneficial for creating individualized training or assistive tools.

We propose to use a basic reinforcement learning model capable of high level performance in a relatively complex, dynamic task as a means of establishing the ground truth of quality for individual decisions. By using the model's feature weights to determine not the optimal action, nor the total number of actions, but rather the number of reasonable actions, we hope to dynamically capture the difficulty of individual decision points and compare how the decision space changes across learners with increasing expertise levels.

Measuring Decision Difficulty

The performance of an individual is significantly impacted by the subjective degree of difficulty of the task at hand. While difficulty has long been considered as a factor when studying decision making, the classification of difficulty is usually determined by the researchers, is coarse-grained in scope (e.g., single digit multiplication compared to double digit multiplication), and applied universally to all subjects. These studies often identify effects associated with difficulty, but assume a uniformity of experience across task items and subjects that does not reflect the reality of a particular individual. For complex or dynamic tasks, identifying the "right" answer at any given decision point becomes challenging, making a quantitative assessment of difficulty almost impossible — the correct action may change as a result of the task environment, or there may be no objectively best action available. Performance in such tasks is a cumulative result of all actions, each with their own level of difficulty. It is precisely these tasks, without objective correct choices made in dynamic environments under time pressure, where AI support systems have great potential for instruction and support of highly skilled users. But for that support to be successful, it will need to be possible to identify how an individual user experiences the difficulty of each task action.

In an effort to begin exploring this challenge, we use AI models that are capable of playing the video game Tetris, a complex, dynamic, decision-making task, at a high level of performance. When playing Tetris, human players make decisions about where to place a sequence of geometric shapes (called "zoids") into a pile on the board such that the pieces form unbroken lines across the full width of the screen. When this is accomplished, the lines vanish and the player earns points. This continues, with the players having less and less time to make their choices as the level speed increases, until the height of the pile reaches the top of the screen. The AI models perform the same task by using a set of weighted features of the board (such as pile height, or number of unfilled squares) to score all possible placements at each decision point and choosing the position with the highest score. Various machine learning methods have been used to train optimal feature weights that achieve final game scores similar to human experts ((Thiery & Scherrer, 2009b; Szita & Lorincz, 2006; Sibert, Gray, & Lindstedt, 2015)).

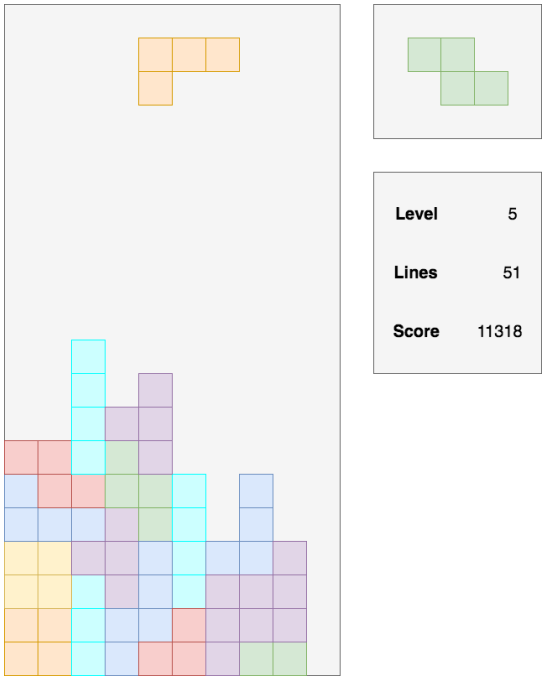


Figure 1: A typical episode in Tetris

Tetris makes an excellent task for studying decision making because the environment is fairly simple, but there is a large space of possible decision configurations. Additionally, no single decision in a game of Tetris is critical to the overall performance; skilled players are often able to quickly recover from what appear to be catastrophic errors. This level of performance is not the result of learning optimal state-action pairs, as players are unlikely to encounter exactly the same decision space twice, but instead by developing a decision policy that uses information in the environment to identify a good choice in any possible decision configuration. By getting a high score, human and model players alike demonstrate

that their policies are effective.

Most of the time, however, the focus on models in decision making has been on what they do: the decision they ultimately make. But what the models don't do can also be informative about the decision making space. In order to make a decision about which position to select, the AI models use their optimized feature weights to select the highest scoring move. In doing so, they also generate a score for all other potential placements, making it possible to explore the context in which the decision is made, and to gauge the difficulty of choosing between the available options. Here, using the expectation that more available moves might mean that decisions are harder, as popularized by the famous Hick's Law (Hick, 1952), we propose a new metric for estimating decision difficulty.

Methods

A preexisting model that plays Tetris at an expert level (Sibert et al., 2015) was used to evaluate the decision space of a large set of decisions made by humans across a wide range of skill levels. The number of options deemed "reasonable" was mapped to the difficulty of that decision.

Models of Tetris

The possible decision space of Tetris is extremely large. This means that expert players are not learning optimal state/action pairs for specific board states in order to be successful. Instead, they develop a sense of the "goodness" of particular placement options based on the available visual board information. This process can be approximated with machine models by identifying definable features of the board, such as height or the number of unfilled and inaccessible cells (called "pits"), and evaluating possible placements based on how they affect the values of those features. Weights can be assigned to each feature that reflect their relative importance to the decision making process.

At each decision point, the model generates all possible placements for the active zoid, multiplying the value of each board feature that results from each candidate placement by the associated weight of that feature. These values are then summed to produce a numerical score for each placement. The model selects the move with the highest score, and this becomes the new decision space for the next placement or trial in the game. This process repeats until the game ends.

Various machine learning methods can be applied to find the optimal set of weights that produce high-level performance in Tetris, and indeed, Tetris has been used as an example case for demonstrating the effectiveness of methods for searching large, uncertain feature spaces (Robertson, 2012; Carr, 2005; Boumaza, 2009; Thiery & Scherrer, 2009a). However, while these models are capable of achieving almost unbelievably high scores, they are not particularly helpful for explaining human behavior because they omit one of the most critical features of the game as humans play it: time pressure. Without time pressure, the models adopt a significantly different strategy from humans who play under limitations, and therefore what makes a particular placement "good" will differ for each.

It is possible to train models to behave more like humans. Even a simple approximation of time pressure in the form of a limited game length produces models with a behavior pattern much closer to that of highly skilled human players (Sibert et al., 2015), and presumably, a similar sense of what makes a move "good" or not.

The model used in this study was adapted from Sibert (2015). Models were trained using Cross-Entropy Reinforcement Learning, a genetic-algorithm-like method of reinforcement learning that narrows down the search space, in this case of feature weights, over multiple generations (Thiery & Scherrer, 2009a, 2009b; Szita & Lorincz, 2006). Optimizing for score, models were trained until the weights of the best performing models in a generation converged, usually between 30 and 40 generations.

Measures of Reasonableness

When exploring the behavior and strategy of the model, the only placement score that really matters is the highest one. However, in order to make its decision, the model must assign a numerical score to all possible placements at each decision point (between 9 and 34 options, depending on the zoid). Humans do not consider all possible placements, and, in fact, may not be considering more than one option most of time. Simsek (2016) suggests that, even without explicit weights, a simple ranking policy of features is capable of identifying a clear best move in most Tetris decisions. But even if "most decisions in Tetris are easy", it is less obvious what it means for one placement to be a clear winner, and what is happening in the cases where there isn't one.

To better explore these questions, we used the weights from the model described above to assign a numerical score for all possible placements available during each of over 500,000 decisions made by human players. Using the highest scoring move as a reference point, we computed the distance of all other other candidate moves from the reference point. The standard deviation of these distances was used to create a criterion point for that decision space. Moves that fell between the high score and the criterion point were considered "reasonable", that is, maybe not precisely the highest, but, according to the model's sense of move goodness, still a good choice. Moves with distances greater than the criterion point were considered "unreasonable" and unlikely to be seriously considered.

Using different standards for setting the criterion point, we were able to determine the proportion of moves with only a single reasonable move (i.e. easy moves), as well as identifying moves of increasing difficulty based on the number of plausible options (Figure 2).

Results

The models assigned scores to over 500,000 individual decisions made by 500 participants in the video game Tetris,

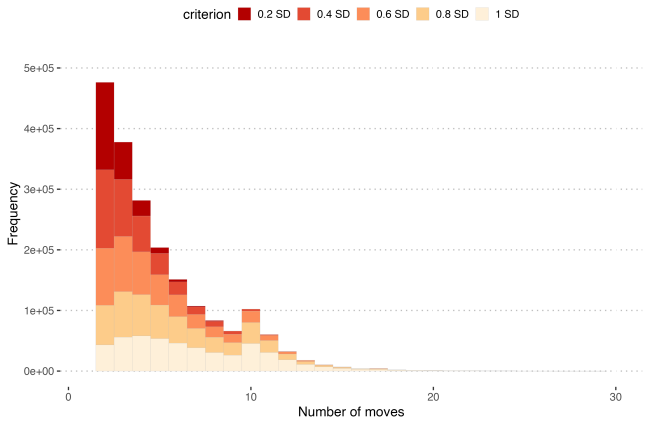


Figure 2: Histogram of moves suggested by model for all decision points following inclusion criteria on distance of rating from the optimal move, from most permissive (beige) to most restrictive (dark red). SD is standard deviation.

and using various criterion points, the number of "reasonable" moves were determined for each decision point. Participants were assigned to skill bins based on the average of the four best scores they achieved during an hour of play time: Extreme Novice, Novice, Beginner, Intermediate, Proficient, Advanced, Expert, and Extreme Expert. These groups are used to compare decision difficulty as measured by the model, with human skill level.

Model Assessment of Decision Difficulty

The model rated and rank-ordered all available moves at all decision points, the first ranked move being the optimal move for that given Tetris board configuration. As is typical of dynamic decision making tasks, the previous decisions made by the players resulted in different board configurations, meaning that it is very unlikely that two players will ever encounter the exact same decision space. As such, while it is possible that a few board states (such as those at the beginnings of games) are repeated and are thus overrepresented in the set, each board configuration is treated as a unique decision space. In this analysis, we counted the number of optimal moves the model suggests as a measure of difficulty for specific decision points. On average there were 23.1 available moves (*SEM* = 0.207) identified by the model for each decision point across the individual games. There is always at least one optimal move that has the highest rating, and there may be other candidate moves that have the same score, and still others that might score lower, but so close as to be effectively the same. At some point, however, the moves are rated too low to be considered viable. To determine this cutoff point, we devised inclusion criteria (see Figure 2) with varying levels of strictness to count the most optimal moves: the most permissive criterion counts move ratings that are one standard deviation (SD) away from the top ranked move as being reasonable, while the most restrictive criterion includes only those moves that are with in 20 percent of the 1 SD (stepping down in 20 percent increments). As a middle ground, we selected 0.6 SD as our inclusion criteria in the following analyses, as 0.2 SD is too restrictive and 1 SD is too permissive.

Table 1: Mean number of optimal moves by criterion. SEM is standard error of the mean.

criterion	Mean	SEM
0.2 SD	2.231	0.115
0.4 SD	3.529	0.177
0.6 SD	4.892	0.215
0.8 SD	6.146	0.233
1 SD	7.413	0.237

We found that, at a 0.6 SD cut-off, there are on average 4.89 reasonable moves at each decision point (*SEM*= 0.215) across games (the most permissive 1 SD cut off results in an average of 7.41 optimal moves (*SEM* = 0.237); see table 1 for counts as a result of the other cut-off criteria). This suggests that most decisions in Tetris have, on average, between 4 and 5 optimal moves, 2 being the most common (this is a positively skewed distribution), and are therefore not necessarily 'easy'. While decision points with only one optimal move are frequent, they do not represent an overwhelming majority. Decision points with very high numbers of optimal moves (greater than 10) are infrequent.

To examine how the decision difficulty varies by expertise, we compared the amount of optimal moves across skill bins, taking the average of optimal moves per player, in all the games they played. There were few to no differences between skill groups, but the Extreme Novice players had higher counts of optimal moves (and therefore more difficult decisions) on average compared to the other groups (see Figure 3). However, we have too few players (n=4) in this group, to say anything with confidence.

Next, we asked the question - in all of the decisions that players made, what was the proportion of low numbers of optimal moves, like 1 or 2, to higher numbers of optimal moves? And, how does this differ across skill groups? Lower numbers of optimal moves would suggest easier decisions, and larger numbers would suggest increased difficulty. We computed the proportion make-up of counts of optimal decisions for each decision point and compared them across players in the different skill groups (Figure 4). We found that decisions made by experts (cooler colors in Figure 4) had slightly higher proportions of *low* counts of optimal moves (more frequent moves of easier difficulty), and this trend reverses as we move towards higher counts of optimal moves (i.e., players with lower skill levels, warmer colors in Figure 4, tended to have higher proportions of decision made up by large counts of optimal moves, and therefore more difficult decisions).

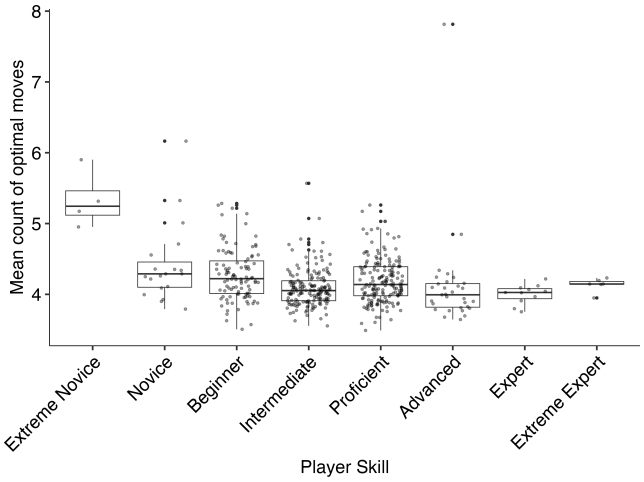


Figure 3: Box-plots of mean number of optimal moves as found by the model, for all decision points, broken up by players' skill level.

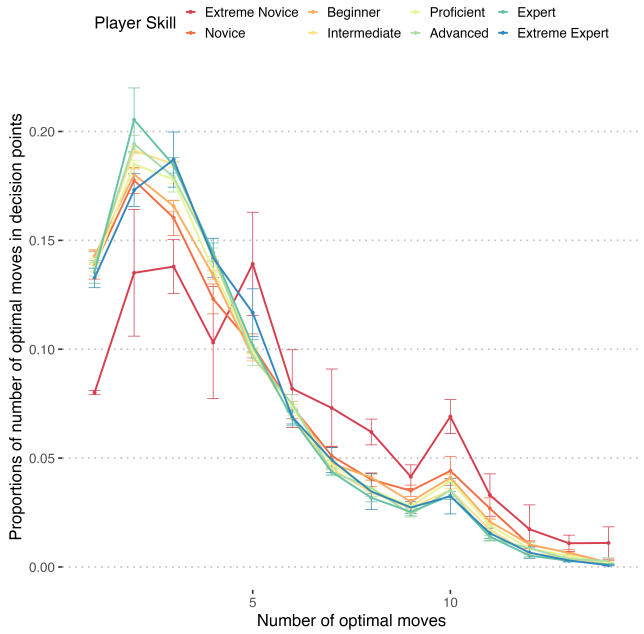


Figure 4: This figure shows the average numbers of optimal move counts that make up each decision point broken up by player expertise level. For example, only about 8 percent of decisions made by Extreme Novice players (dark red) had one optimal move. Error bars show standard errors across players for each skill group.

Predicting players' subjective experience of difficulty

Given the results above, we next sought to test if model-assessment of decision difficulty is reflected in the subjects' behavior. In this analysis, assuming that more difficult decisions might take more time, especially if players are exploring a decision space with a larger number of possible options, we used a linear mixed-effects model (lme4 package for R (Bates, 2010)) to find the relationship between counts of optimal moves and players' initial latency (i.e., the amount of time players take to press any button at the beginning of a decision epoch). We constructed a main model (initial latency number of optimal moves + player skill) that includes the number of optimal moves and player skill since we have established that skill level affects the progression of the games and was a variable of interest to us. Other factors that might influence our results, and were repeated across subjects, were entered in to the model as random factors (Game, decision number, and subject number). For comparison, we built two additional models, the second without the expertise factor (initial latency number of optimal moves) and the third, without the optimal move count (initial latency player skill), to establish the individual contributions of those factors to the initial latency measure.

After fitting the models we used Bayesian Information Criterion (BIC; (Schwarz, 1978)) to determine model fit, and rank models through relative comparison of the models by taking the best fitting model of the three as the reference ((Raftery, 1995). We found that the full model was significantly better fit to the data in predicting initial latency compared to the other two models and so this model was taken as the best fit model. The second best fit model contained only the expertise factor and had a difference $\Delta\text{BIC} = 9$, which suggests that the full model has strong evidence (Kass & Raftery, 1995) over the expertise only model. This difference was much larger when compared with the third model which contained only the number of optimal moves factor, $\Delta\text{BIC} = 111$, which suggests very strong evidence for the full model against the third model (number of moves only). Note that the same random factors were entered into all of the models.

Discussion

When exploring the full decision space of Tetris, we find that, while many decisions can be considered easy (i.e. there is a single, obvious best placement to make), easy moves do not make up a dominant proportion of all moves examined. By introducing a flexible method for determining move reasonableness based on move scores calculated by an AI model, we are able to classify individual decision points by difficulty level. Overall, it is most frequently the case for a decision to have three or more good options. Truly difficult moves, with 10 or more options, are very infrequent.

Examining the relationship to expertise level, we found that there were modest differences among players of different skill levels. Decision points in games played by extreme novice players had more optimal moves associated with them compared to other groups. There was a slight trend to decreasing amounts of optimal moves as expertise increases, which suggests that experienced players manage the game board more optimally, and as a result, have prepared good potential

placements for any upcoming moves. This aligns with self-described strategies of expert level players. Novices, lacking this forethought, may be dealing with boards that are not well prepared for upcoming pieces, and must face decisions with many relatively poor placement options. This trend, however, does not hold as we progress from Advanced to Extreme Expert players, which suggests that these players might be using strategies that our model is not aware of and they might be out-performing the model. Small numbers of optimal move options were associated with a higher proportion of decisions made by experts compared to novices, though the difference is relatively modest.

While applying this metric has given us an initial window into the difficulty of decisions in Tetris, many questions remain. First, the difficulty classification is based entirely upon the model's evaluation of the available moves, and it is not yet determined if this has any connection with the subjective experience of difficulty that a particular player might have. The statistical results of the model that uses the number of options as a predictor of initial latency do suggest a relationship between our measure of difficulty and decision time, and this is a promising first indication that the difficulty rating is reflected in observable player behavior. However, this initial relationship must be more thoroughly investigated before a definitive connection can be proposed.

Second, the model plays at the level of an expert player, and as such, difficulty is measured by the number of moves that an expert considers to be reasonable. However, novices and experts may differ in what they consider a good move to be, which might change the number of moves classified as worth considering, and thus the difficulty of decisions. This may account for the lack of difference between the relative difficulty of moves across expertise groups. The current model is trained to optimize game score, but incorporates no human data directly. Furthermore, we see a pattern where decisions points for extreme experts lie on the fringes of what the model can predict about them, which might come down to the limited solving strategies of the Tetris model. The models represent a single decision policy, which can be somewhat flexible, but generally follows one strategy. Extreme expertise may be better categorized by the ability to switch between drastically different strategies that are not possible to represent with the current model structure. It is possible to train models to match human decisions, therefore, and it may be possible to build models that play at each of the eight skill levels. The hope is that these models would better reflect what is considered "good" at each level of expertise, and produce a more accurate judgment of the difficulty of each decision point.

Despite these limitations, these results demonstrate that a simple model of a complex task has great potential for providing insight into the nuances of a complex, dynamic task. Accurately identifying points of higher and lower difficulty will allow AI systems to provide support and assistance only when needed, and reduce distractions caused by unnecessary interventions. By using the model to segment the data by the number of good moves, we hope to develop a method for robustly identifying difficulty for specific individuals, both in Tetris and in similar tasks.

References

Ashby, F. G., & Maddox, W. T. (2011). Human category learning 2.0. *Annals of the New York Academy of Sciences*, *1224*(1), 147–161.

Bates, D. M. (2010). *lme4: Mixed-effects modeling with r.* Springer.

Boumaza, A. (2009). On the evolution of artificial tetris players. In *2009 ieee symposium on computational intelligence and games* (pp. 387–393).

Brehmer, B. (1992). Dynamic decision making: Human control of complex systems. *Acta psychologica*, *81*(3), 211–241.

Carr, D. (2005). Applying reinforcement learning to tetris. *Department of Computer Science Rhodes University*.

Collins, A., & Koechlin, E. (2012, 03). Reasoning, learning, and creativity: Frontal lobe function and human decision-making. *PLOS Biology*, *10*(3), 1-16. Retrieved from https://doi.org/10.1371/journal.pbio.1001293 doi: 10.1371/journal.pbio.1001293

Collins, A. G. (2018). The tortoise and the hare: Interactions between reinforcement learning and working memory. *Journal of cognitive neuroscience*, *30*(10), 1422–1432.

Gläscher, J., Daw, N., Dayan, P., & O'Doherty, J. P. (2010). States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, *66*(4), 585–595.

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

Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of experimental psychology*, *4*(1), 11–26.

Hutton, R. J., & Klein, G. (1999). Expert decision making. *Systems Engineering: The Journal of The International Council on Systems Engineering*, *2*(1), 32–45.

Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., ... Kasneci, G. (2023). Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, *103*, 102274. doi: 10.1016/j.lindif.2023.102274

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the american statistical association*, *90*(430), 773–795.

Liévin, V., Hother, C. E., Motzfeldt, A. G., & Winther, O. (2024). Can large language models reason about medical questions? *Patterns*, *5*(3), 100943. doi: 10.1016/j.patter.2024.100943

Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological methodology*, 111–163.

Robertson, J. (2012). *A brief history of tetris ai.*

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 461–464.

Shanteau, J. (1988). Psychological characteristics and strategies of expert decision makers. *Acta psychologica*, *68*(1-3), 203–215.

Shen, Y., Song, K., Tan, X., Zhang, W., Ren, K., Yuan, S., ... Zhuang, Y. (2023). Taskbench: Benchmarking large language models for task automation. In *Advances in neural information processing systems* (Vol. 37, pp. 4540–4574).

Sibert, C., Gray, W. D., & Lindstedt, J. K. (2015). Tetris-^{TM}: Exploring human performance via cross entropy reinforcement learning models. In *Proceedings of the 37th Annual Conference of the Cognitive Science Society* (pp. 2188–2193).

Şimşek, Ö., Algorta, S., & Kothiyal, A. (2016). Why most decisions are easy in tetris—and perhaps in other sequential decision problems, as well. In *International conference on machine learning* (pp. 1757–1765).

Szita, I., & Lorincz, A. (2006). Learning Tetris using the noisy cross-entropy method. *Neural Computation*, *18*(12), 2936-2941.

Thiery, C., & Scherrer, B. (2009a). Building controllers for tetris. *Icga Journal*, *32*(1), 3–11.

Thiery, C., & Scherrer, B. (2009b). Improvements on learning Tetris with cross-entropy. *International Computer Games Association Journal*, *32*(1), 23-33.

Simulating Human Cognition in Abacus Gesture Learning: An ACT-R with Vision/Motor and PyIBL Approach

Lingyun He (iris.he@psu.edu)
Department of Industrial and Manufacturing Engineering,
The Pennsylvania State University, University Park, PA 16802 USA

Farnaz Tehranchi (farnaz.tehranchi@psu.edu)
School of Engineering Design and Innovation
The Pennsylvania State University, University Park, PA 16802 USA

Abstract

This paper investigates the cognitive processes for abacus gestures (a set of mid-air gestures representing numbers 0-99). We developed multiple cognitive models using ACT-R architecture with vision and motor modules and PyIBL to simulate learning process. By enhancing previous research models with VisiTor (Vision + Motor) framework, we added visual attention and motor movements involved in abacus gesture learning. Our results demonstrate that ACT-R models can show the learning curve with the contribution of the retrieval processes. In PyIBL, we decomposed each gesture trial into multiple decision sets to reflect human-like gesture presentation process. This structure enabled the model to produce the learning curve with increasing reward over time. This work discovers the retrieval processes, rather than vision or motor modules/knowledge, dominate the learning curve in abacus gesture learning processes. The findings through cognitive modeling provide insights into gesture-based interaction design and human cognition during mid-air gesture learning.

Keywords: Abacus Gestures, Cognitive Modeling, ACT-R, PyIBL, Vision, Motor

Introduction

Gesture control technology has revolutionized the understanding of humans, human cognition, and human-computer interaction by allowing users to control devices through hand movements instead of traditional input methods like keyboards and mouse (Bhuiyan & Picking, 2009). Gesture-based interfaces that let users control devices with, for example, hand or finger motions are becoming increasingly popular. This technology uses sensors and cameras to detect and interpret gestures (i.e., hands and body), allowing users to interact with interfaces and types in the air (Saffer, 2008; Zabulis et al., 2009). Gesture control has been integrated into various fields, such as the automotive industry (Graichen et al., 2022) by providing a safe alternative to in-vehicle interactions and gaming industries (Ionescu et al., 2012) through allowing players to engage with games through physical movements, making gameplay more immersive and interactive (Bhuiyan & Picking, 2009).

Given the significance of gesture control technology across various fields, it is important to comprehend the associated behavior and cognition (Greene et al., 2013). Additionally, simulating these knowledge in modeling approaches is essential for understanding user actions, decision-making processes, and learning mechanisms (Chen & Fang, 2014). Similar modeling efforts in robotics, reinforcement learning, and multi-modal cognition highlight the broader impact of cognitive modeling in complex, human-centered systems (Li et al., 2024; Zhang et al., 2025; Zhao et al., 2025). In this study, we build on prior work using Abacus Gestures—a set of mid-air hand gestures that represent 100 numbers from 0 to 99 through various combinations of open and closed finger—to explore gesture learning from a cognitive modeling perspective (Ehtesham-Ul-Haque & Billah, 2023; He et al., 2024). Figure 1 right shows one way of representing 26 using abacus gestures.

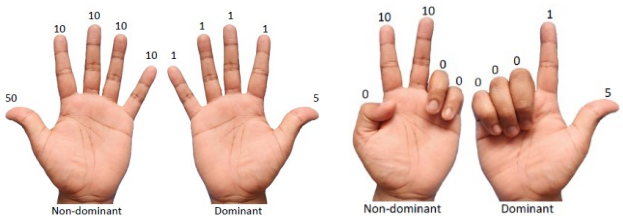


Figure 1: Abacus Gestures (Ehtesham-Ul-Haque & Billah, 2023). *Left*: The set of numbers assigned to 10 fingers in the abacus gesture. *Right*: The abacus gesture represents 26 by opening the index and middle fingers of the non-dominant hand and the thumb and index fingers of the dominant hand.

In this work, we investigate the cognitive processes of abacus gesture learning using ACT-R with vision and motor modules and PyIBL. We focus on modeling motor and visual perception in ACT-R, and how the decomposition of gesture trials into human-like decision sets influences learning performance in PyIBL. This paper aims to answer the following research questions:

(1) Can an ACT-R model with vision and motor knowledge predict human behavior when using abacus gestures to represent the numbers 0-99?

(2) Do human-inspired decompositions of behavior improve learning in instance-based models?

(3) What factors contribute to the process of learning abacus gestures, perhaps including visual search and motor movements?

These questions are motivated by the need to evaluate both perceptual-motor alignment (via ACT-R) and memory-based learning effects (via PyIBL). To address these questions, we developed multiple cognitive models using cognitive architecture: 1. ACT-R (Anderson, 1990; Anderson & Lebiere, 1998; Ritter et al., 2019) along with VisiTor (Bagherzadeh & Tehranchi, 2022). 2. PyIBL (Gonzalez et al., 2003). This study aims to investigate whether retrieval-based cognitive mechanisms are sufficient to explain gesture learning dynamics, and whether incorporating human-like action decomposition improves learning performance in instance-based models.

Cognitive Models

Cognitive models predict an individual's behavior and cognitive processes by modeling human knowledge, memory, and reasoning, providing critical insights into cognitive processes (Newell, 1990; Ritter et al., 2019). Cognitive architecture is the foundational framework of the cognitive model and defines the basic structure underlying cognitive processes. For this work, we selected ACT-R (Anderson, 1990; Anderson & Lebiere, 1998; Ritter et al., 2019) and PyIBL (Gonzalez et al., 2003) due to its comprehensive framework, including vision and motor tools like VisiTor (Bagherzadeh & Tehranchi, 2022).

ACT-R

ACT-R has two types of human knowledge: declarative knowledge, representing what we know, and procedural knowledge, representing the actions we can perform (Bothell, 2017). ACT-R can model complex learning curves and optimize both short-term and long-term learning processes (Rasmussen, 1986; Tehranchi, 2021; VanLehn, 1996). The ACT-R framework enables the simulation of response times, accounting for factors such as chunk retrieval time and activation values that reflect human memory and learning patterns. In ACT-R, the retrieval time is given by this equation:

$$Time\ (s) = Fe^{-(f*A_i)}$$

with three parameters: activation value (A_i), latency factor (F, default = 1 second), and latency exponent (f, default = 1 second) for chunk i.

Additionally, ACT-R's perception (vision) and action (motor) components allow simulation of eye movements and motor movements such as "peck" for finger motion. In ACT-R, the motor module incorporates several movement styles (specification of a movement type) based on EPIC's Motor Processor (Bothell, 2017; Kieras & Meyer, 1996). ACT-R facilitates the comprehensive integration of cognitive processes, allowing for the simulation of human cognition and behavior (Tehranchi & Ritter, 2017, 2018).

VisiTor

To develop complete and accurate cognitive models that simulate human behavior, the ability to interact with task environments across a range of tasks is required (Byrne, 2012; Kieras & Meyer, 1996). To address the ACT-R's limitation in interaction with dynamic task environments, we utilize the VisiTor (Vision + Motor) framework (Bagherzadeh & Tehranchi, 2022). VisiTor enhances ACT-R's interaction capabilities with task environments through two modules: Vision and Motor. This tool simulates users' visual attention (Vision) and use of mouse and keyboard (Motor). This tool enables the model to interact with the interface as a user would. VisiTor keeps operations as close to user behavior as possible, building on cognitive architectures like ACT-R and PyIBL. Importantly, VisiTor's capabilities are flexible and expandable to accommodate the new interaction requirements.

PyIBL

Instance-based learning theory is known as memory-based learning (Learning, 1997). It provides a flexible and intuitive approach to develop cognitive models. PyIBL (Python Implementation of Instance-Based Learning) implements a computational model of decision-making grounded in memory retrieval processes (Dutt & Gonzalez, 2012; Gonzalez et al., 2003). Unlike ACT-R, which integrates multiple cognitive modules, PyIBL focuses specifically on how decisions are made by retrieving prior experience based on similarity and reward and applying them to new situations.

Modeling Approach

Method Overview

We develop cognitive models using two frameworks: ACT-R and PyIBL, to simulate the abacus gesture learning process. ACT-R models are built using vision and motor modules based on the previous work (He et al., 2024) to simulate vision and finger movement during gesture execution. PyIBL models focus on how different ways of structuring decision sets affect the agent's learning process over repeated training. While ACT-R includes a built-in utility learning mechanism, we select PyIBL to model retrieval-based learning because it allows more flexible and interpretable decision sets.

ACT-R + VisiTor

In previous research (He et al., 2024), we developed three cognitive models (i.e., Model 1, Model 2, and Model 3) using ACT-R (ACT-R 7.26) to simulate the human behavior when learning abacus gestures (He et al., 2024). Reported Model 1 hypothesized that participants possess 100 chunks (abacus gestures from 0 to 99) in declarative knowledge. Model 2 hypothesized that participants possess 19 chunks (abacus gestures from 0, 1, …, 9 and 10, 20, …, 90) and express the gestures through combinations of these chunks. Model 3 hypothesized that participants possess 14 chunks (0, 50 for non-dominant thumb, 0,10,…,50 for non-dominant other

fingers, 0, 5 for dominant thumb and 0,1,…,5 for dominant other fingers), categorizing fingers into the dominant thumb, dominant other fingers, non-dominant thumb, and non-dominant other fingers. 14 chunks can represent 99 abacus gestures through their combination. We used the Boolean flag for each finger to represent their closed and opened states.

In this work, we have created new cognitive models based on reported Model 2 and Model 3 (He et al., 2024) since they are correlated with human data closely. Unlike the previously proposed ACT-R models, the new model adds vision modules using VisiTor (Bagherzadeh & Tehranchi, 2022) to simulate the process for the shifts of attention in the human mind. In addition, we introduced the motor module and manual knowledge in the new models. Figure 2 shows the original numerical display window used in the abacus gesture study with the original Python code (Ehtesham-Ul-Haque & Billah, 2023) and the new digital display window we developed in Python to interact with ACT-R models.

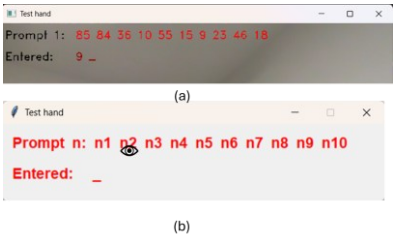


Figure 2: (a) Numerical display window used in the abacus gesture study from Ehtesham-Ul-Haque et al. (2023). (b) Numerical display window used in ACT-R with VisiTor eye icon.

For the new models proposed in this paper, we simulated the reading process using the vision and motor module of ACT-R. In the previous abacus gesture study (Ehtesham-Ul-Haque & Billah, 2023), a specific task window showed 10 numbers at a time. This is the setting in the human experiment, and we simulate it to restore the same human experiment as much as possible. After completing the task of ten numbers, the task window showed ten new random numbers. In ACT-R simulation, we simulate this process by reading ten placeholders for numbers at a time. For instance, the model reads “n1”, and a random number in Lisp will be assigned to “n1”. After the model finish reading ten random numbers and representing their corresponding abacus gestures, the model shifts attention to the “n1” in order to read these placeholders for numbers again in the new set. This process continues until all abacus gesture tasks are completed, which consist of ten rounds, with each round involving the execution of ten individual numbers. We use VisiTor to simulate eye-movement for visual attention (the eye icon in Figure 2 is VisiTor attention area). The “Prompt” field displays the target number, while “Entered” indicates the participant's response using abacus gestures.

We use the "Peck" function to effectively simulate a specific raising of the finger. ACT-R has different motor movement styles such as “Punch”, “Peck”, “Peck-recoil”, “Ply”, and “Hand-ply” (Bothell, 2017). "Punch" involves a simple downstroke followed by an upstroke, and "Ply" and "Hand-ply" are designed for device movement, “Peck” simulates targeted finger positioning by executing a directed movement at a specified distance and angle. It closely simulates the process of raising a finger to a specific position in space, which is essential for mid-air gestures. Additionally, "Peck-recoil," automatically returns to the starting position. We use "Peck" to maintain the finger in its raised position until the model finishes the gestures. “Peck” function needs the input angle and distance. We set the distance as 1 and the angle to 90 degrees if the model is opening the finger. If the model is closing the finger, the angle is set up as - 90 degrees. Figure 3 shows the sample production rule for motor module that opens the finger for the “peck” function.

```
  !output!       ("Open Non-dominant Thumb")
+manual>
    cmd     peck
    hand    left
    finger  thumb
    r       1
    theta   90            )
```

Figure 3: Sample motor module that shows thumb on the left hand (i.e., non-dominant hand) has been raised.

Model A builds on reported Model 2 (He et al., 2024) with the same declarative (chunk) structure, and Model B is extended from the reported Model 3 (He et al., 2024) with the same logic of declarative knowledge. For Model A, we set the model to raise fingers in the sequence from thumb to little finger, first on the non-dominant hand and then on the dominant hand. Also, the fingers are closed one by one in the same order. The flowchart of Model A is shown in Figure 4.

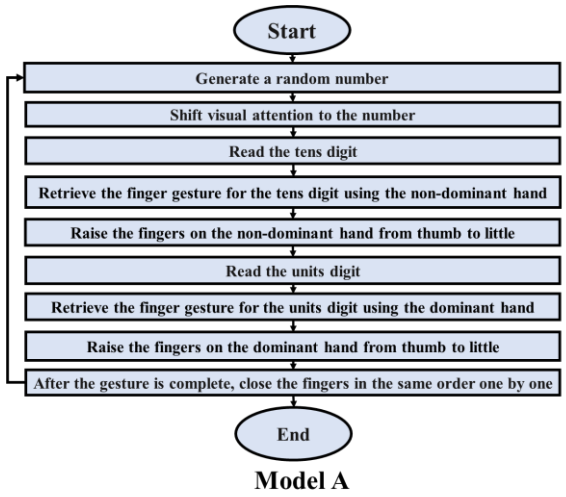


Figure 4: Workflow for ACT-R Model A

For Model B, there are four finger types based on non-dominant vs. dominant hands and thumb vs. non-thumb fingers (as in Model 3 from He et al. (2024)). We simulate this behavior by first opening the non-dominant hand's thumb,

then opening the remaining fingers of the non-dominant hand (only using a single "peck" with index since we assumed the other fingers open together) and repeating the same process for the dominant hand. After that, we do the same sequence for closing the raised fingers. The Flowchart for Model B is shown in Figure 5, demonstrating the necessary cognitive steps to complete the abacus gesture learning task.

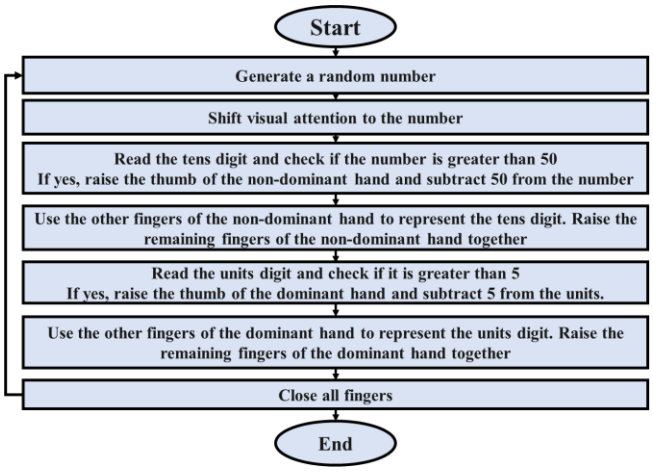


Figure 5: Workflow for ACT-R Model A

PyIBL

PyIBL focuses on modeling decisions based on past experiences, which are stored as instance: individual records of prior decisions and outcomes. We focus on a specific decision sets within a gesture trial that can lead to a reward. The logic of our implementation is developed based on the previous research's findings (He et al., 2024).

Three PyIBL models are developed:

PyIBL Model 1: The PyIBL agent selects the correct options from all combinations of fingers. If one of the correct combinations is selected, a reward of 10 for the agent will be returned. For example, if the random number is 1, the correct choices should be (1, ('dominant-index')), (1, ('dominant-middle')), (1, ('dominant-ring')) or (1, ('dominant-little')).

PyIBL Model 2: The PyIBL agent needs to make 2 choices. The first one is to select the correct option among all combinations of the non-dominant fingers, and the second one is to select the correct option among all combinations of the dominant hand fingers. Each correct choice returns a reward of 5. There are two choices, so the total reward is 10. For example, if the random number is 1, the correct choice for the non-dominant hand fingers should be (1, ()) and the correct choice for the dominant hand fingers should be one of the (1, ('dominant-index')), (1, ('dominant-middle')), (1, ('dominant-ring')) or (1, ('dominant-little')).

PyIBL Model 3: In order to improve the learning efficiency of the model 2, we reduce the number of options in the decision set at each time step. The agent makes 4 choices each round for non-dominant thumb, non-dominant other fingers, dominant thumb, and dominant other fingers separately with the given random number. In each choice, if the PyIBL agent gives the correct choice from the combinations of fingers, then it will receive a reward of 2.5. If all the choices are correct, the total reward is 10. For example, if the random number is 1, the correct choice for the non-dominant thumb fingers should be (1, ()). The correct choice for the non-dominant other fingers should be (1, ()). And the correct choice for the dominant thumb fingers should be (1, ()). The correct choice for the dominant other fingers

Table 1: Number of choices the agent makes for one abacus gesture, decision set and example (random number =1) for PyIBL Model 1, 2, 3

Model	Number of choices	Decision Set	Example (if random number = 1)
Model 1	1	One decision: Choose one combination (including an empty set) from all possible finger configurations.	(1, ('dominant-index'))
Model 2	2	Decision 1. Choose one combination (including an empty set) from all non-dominant finger configurations. Decision 2. Choose one combination (including an empty set) from all dominant finger configurations.	Non-dominant: (1, ()) Dominant: (1, ('dominant-index'))
Model 3	4	Decision 1. Choose one combination (including an empty set) for non-dominant thumb. Decision 2. Choose one combination (including an empty set) for non-dominant other fingers. Decision 3. Choose one combination (including an empty set) for dominant thumb. Decision 4. Choose one combination (including an empty set) for dominant other fingers.	Non-dominant thumb: (1, ()) Non-dominant 4 other fingers: (1, ()) Dominant thumb: (1, ()) Dominant 4 other fingers finger: (1, ('dominant-index'))

should be one of the (1, ('dominant-index')), (1, ('dominant-middle')), (1, ('dominant-ring')) or (1, ('dominant-little')). Model 3 utilizes a more realistic cognitive decomposition, where humans separately assess finger groups based on hand dominance and finger roles, reducing the cognitive load and limiting the decision set.

Across all three models, we systematically reduce the number of choices in decision set by decomposing a single gesture into different decision points, allowing us to test whether cognitive-inspired decision sets facilitate faster reward-based learning. Table 1 summarizes how each PyIBL model defines its decision structure and the corresponding choice set.

Results

ACT-R + VisiTor

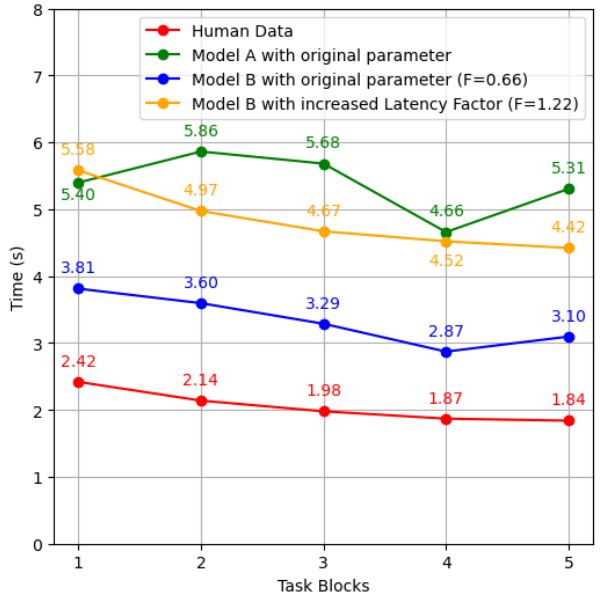


Figure 6: Average time for making an abacus gesture of Model A, Model B and Model B with increased Latency Factor are compared with human data, showing the learning patterns in making abacus gestures.

We initially utilize the same parameters as the previous Model 2 and Model 3. VisiTor is responsible for modeling visual attention, while ACT-R handles finger movement. However, the result does not show an accurate learning curve, even though Model B performs slightly better than Model A. Model B shows the learning curve for the first 4 blocks and has better correlation value with human data, which is 0.938 while Model A's correlation value is 0.398. One possible explanation for this difference in performance is the number of the "peck" functions in each model. More "peck" functions mean that the motor module takes more time, making it harder to get the learning curve. Model B only uses one "peck" function for all non-dominant other fingers and for all dominant other fingers, Model B also uses the single "peck" function. In contrast, Model A contains multiple peck functions (one "peck" per finger), which means the motor module consumes most of the processing time. This extensive motor processing makes it difficult to demonstrate a learning curve, even as we observe the increasing activation values in the retrieval process. To validate this assumption, we increase the Latency Factor (F) to increase the time required for the retrieval process. The result shows that when the latency factor is increased and the retrieval process becomes the dominant time component, Model B successfully displays the learning curve with a higher correlation value (0.998) even though it takes a longer time. So, in the abacus gesture learning process, the time consumed by vision and motor modules does not dominate the overall process. For instance, visual search in this task does not contribute to the learning process. The learning curve is primarily based on the retrieval time of declarative knowledge. The results are shown in Figure 6, suggesting learning curves for Model A, Model B and Model B with increased Latency Factor across five blocks, each containing 20 abacus gestures for a total of 100 abacus gestures. Table 2 presents the correlation between the model and human data, as well as the corresponding RMSE values.

Table 2: Comparing the fit of learning curves to human data for the abacus gesture learning.

Model	Correlation (Human vs. Model)	RMSE (Human vs. Model)
Model A with Original Parameter	0.398	3.355
Model B with Original Parameter	0.938	1.293
Model B with Increased Latency Factor (F=1.22)	0.998	2.792

PyIBL

We record the total reward for each round. In addition, the average reward is calculated to analyze the performance trend of PyIBL agents over time. We divide the 500 rounds into five equal blocks, with each block consisting of 100 rounds.

Figure 7 presents the average reward per block for the three models. As shown in Figure 7, Model 3 not only outperforms others in final reward but also shows a smooth, upward learning curve. This suggests that Model 3 is effectively improving its decisions over time. In contrast, Model 1 and Model 2 show relatively flat curves, with no meaningful upward trend and learning curve. The results show that human-like actions based on cognitive models' insights can help constrain the choice sets in each round, improving IBL learning curves.

In the 500 rounds, the performance of Model 3 is better than Model 2 and Model 1. To better compare the three models, we also conduct a significant experiment on the last

block (100 rounds) to ensure that the differences between the models are significant. The result is shown in Table 3, suggesting Model 3 is significantly different than Model 1 and Model 2 in 401-500 rounds. It means Model 3 significantly outperforms both Model 1 and Model 2. Model 2 slightly outperforms Model 1.

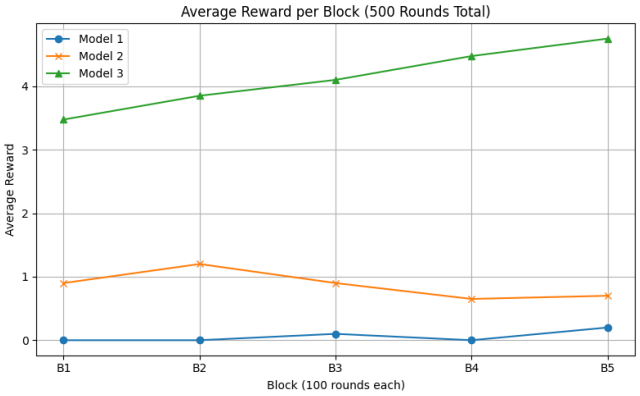


Figure 7: Average rewards for three PyIBL models under 500 different rounds. The 500 rounds were divided into five equal blocks of 100 rounds each. Only Model 3 shows a consistent upward trend.

Table 3: Independent *t-test* comparing PyIBL models on the final block (401 – 500 rounds).

Comparison	t-Statistics	p-Value	Significance
Model 1 vs Model 2	-2.13	0.0346	* ($p < 0.05$)
Model 1 vs Model 3	-13.79	8.97×10^{-31}	*** ($p \ll 0.001$)
Model 2 vs Model 3	-11.48	1.03×10^{-23}	*** ($p \ll 0.001$)

Conclusion

This study investigates the cognitive process of abacus gesture learning through ACT-R with vision and motor modules and PyIBL. Our results show that both ACT-R and PyIBL can model abacus gesture learning process, addressing our initial research questions.

For the first research question, we found that Model B, with a larger latency factor, based on the reported Model 3 with four different finger chunks, produced results that were more consistent with human learning patterns than Model A based on the reported Model 2. The integration of VisiTor's visual and motor modules successfully simulated the attentional and physical aspects of gesture generation. For the second research question—whether human-like action patterns inspired by cognitive modeling enhance IBL learning— we observed that structuring decision steps into constrained decision sets significantly improved PyIBL learning curves. It also confirms the significance of establishing the ACT-R model—providing a framework for other different modeling approaches. This supports the view that insights from cognitive architectures like ACT-R provide a valuable foundation for modeling and guiding decision-based learning. For the third research question on the factors that influence abacus gesture learning, we found that the retrieval process dominates the learning curve, while the vision and motor components play a limited role. This finding is supported by our improvement of the latency factor, which shows that the learning curve becomes clearer when retrieval time becomes the dominant factor.

In conclusion, our cognitive models in ACT-R provide valuable insights into the development of cognitive processes involved in learning and predictions of the performance for abacus gesture learning. Meanwhile, the PyIBL results show the critical role of cognitively inspired decision set in enabling effective reward-based learning. We demonstrate that constraining the choice set in ways inspired by cognitive structure (as modeled in ACT-R) significantly improves instance-based learning outcomes. These findings demonstrate the importance of retrieval-based mechanisms in modeling mid-air gesture acquisition.

Discussion and Future Work

Our study provides several significant insights into the cognitive processes in abacus gesture learning. The comparison between Model A and Model B in ACT-R showed that the retrieval process significantly contributes to the learning curve, while the vision and motor modules have less impact on learning. This finding was validated by increasing the latency factor (F), which increased the time for the retrieval process. The implication is that cognitive processing time, rather than physical execution time, is the primary learning part of gesture learning. From the result, the learning in vision and motor parts has limited impacts on the final results. However, a key limitation remains: ACT-R's vision and motor modules currently lack learning mechanisms, meaning perceptual and motor improvements over time are not captured. Addressing this gap will be our critical direction for future research. Besides, the PyIBL models results show the importance of cognitively inspired action decomposition in mid-air gesture acquisition. The learning curves generated by the PyIBL model suggest that appropriate decision sets are important in training the model to adapt and become proficient in gesture-based interaction systems.

Future work involves developing a new type of knowledge and functionality in the motor module for raising a finger, as the existing "Peck" function only simulates movement along the horizontal axis rather than the vertical. Additionally, the noticeable root-mean-squared error (RMSE), such as 2.792 in the ACT-R models' predictions, indicates that further investigation is needed to ensure the models accurately represent human cognition in this task. Moreover, exploring reinforcement learning as an alternative framework could provide valuable insights into how agents adapt through trial and error, and represent a promising direction for future model development.

References

Anderson, J. R. (1990). *The adaptive character of thought* (1st ed.). Psychology Press. https://doi.org/10.4324/9780203771730

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

Bagherzadeh, A., & Tehranchi, F. (2022). Comparing cognitive, cognitive instance-based, and reinforcement learning models in an interactive task. In *Proceedings of ICCM-2022-20th International Conference on Cognitive Modeling*.

Bhuiyan, M., & Picking, R. (2009). Gesture-controlled user interfaces, what have we done and what's next. *Proceedings of the fifth collaborative research symposium on security*, E-Learning, Internet and Networking (SEIN 2009), Darmstadt, Germany.

Bothell, D. (2017). ACT-R 7 Reference Manual. http://act-r.psy.cmu.edu/actr7/reference-manual.pdf

Chen, N.-S., & Fang, W.-C. (2014). Gesture-based technologies for enhancing learning. The *New Development of Technology Enhanced Learning: Concept, Research and Best Practices*, 95-112.

Dutt, V., & Gonzalez, C. (2012). Making instance-based learning theory usable and understandable: the instance-based learning tool. *Computers in Human Behavior*, *28*(4), 1227-1240.

Ehtesham-Ul-Haque, M., & Billah, S. M. (2023). Abacus Gestures: A Large Set of Math-Based Usable Finger-Counting Gestures for Mid-Air Interactions. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, *7*(3), Article 93. https://doi.org/10.1145/3610898

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

Graichen, L., Graichen, M., & Krems, J. F. (2022). Effects of gesture-based interaction on driving behavior: a driving simulator study using the projection-based vehicle-in-the-loop. *Human factors*, *64*(2), 324-342.

Greene, K. K., Tamborello, F. P., & Micheals, R. J. (2013). Computational cognitive modeling of touch and gesture on mobile multitouch devices: Applications and challenges for existing theory. In *Human-Computer Interaction. Interaction Modalities and Techniques: 15th International Conference, HCI International 2013*, Las Vegas, NV, USA, July 21-26, 2013, Proceedings, Part IV 15.

He, L., Ka, D. H., Ehtesham-Ul-Haque, M., Billah, S. M., & Tehranchi, F. (2024). Cognitive models for abacus gesture learning. In *Proceedings of the 46th Annual Meeting of the Cognitive Science Society*.

Ionescu, D., Ionescu, B., Gadea, C., & Islam, S. (2012, March). Gesture control: a new and intelligent man-machine interface. In *Applied Computational Intelligence in Engineering and Information Technology: Revised and Selected Papers from the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics SACI 2011* (pp. 331-354). Berlin, Heidelberg: Springer Berlin Heidelberg.

Kieras, D. E., & Meyer, D. E. (1996). The EPIC architecture: Principles of operation. Retrieved Feb 23, 2025, from University of Michigan, Department of Electrical Engineering and Computer Science Web site: ftp://www.eecs.umich.edu/people/kieras/EPIC/EPICPrinOp.pdf

Learning, M. (1997). Tom mitchell. *Publisher: McGraw Hill*, 31.

Li, P., Abouelenien, M., Mihalcea, R., Ding, Z., Yang, Q., & Zhou, Y. (2024, May). Deception detection from linguistic and physiological data streams using bimodal convolutional neural networks. In *2024 5th International Conference on Information Science, Parallel and Distributed Systems (ISPDS)* (pp. 263-267). IEEE.

Newell, A. (1990). *Unified theories of cognition*. Harvard University Press.

Rasmussen, J. (1986). *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*. Elsevier Science Inc.

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2019). ACT-R: A cognitive architecture for modeling cognition. *WIREs Cognitive Science*, *10*(3), e1488. https://doi.org/https://doi.org/10.1002/wcs.1488

Saffer, D. (2008). *Designing gestural interfaces: Touchscreens and interactive devices*. " O'Reilly Media, Inc.

Tehranchi, F., Oury, J. D, & E. Ritter, F. (2021). Predicting Learning and Retention of a Complex Task Using a Cognitive Architecture. *Proceedings of the Annual Meeting of the Cognitive Science Society* (pp. 1077-1083).

Tehranchi, F., & Ritter, F. E. (2017). An Eyes and Hands Model for Cognitive Architectures to Interact with User Interfaces. In the Proceedings of *MAICS, the 28th Modern Artificial Intelligence and Cognitive Science Conference*.

Tehranchi, F., & Ritter, F. E. (2018). Modeling visual search in interactive graphic interfaces: Adding visual pattern matching algorithms to ACT-R. *In Proceedings of 16th International Conference on Cognitive Modeling* (pp. 162-167).

VanLehn, K. (1996). Cognitive skill acquisition. *Annual review of psychology*, *47*(1), 513-539.

Zabulis, X., Baltzakis, H., & Argyros, A. A. (2009). Vision-Based Hand Gesture Recognition for Human-Computer Interaction. *The universal access handbook*, *34*, 30.

Zhang, Z., Luo, Y., Chen, Y., Zhao, H., Ma, Z., & Liu, H. (2025). Automated Parking Trajectory Generation Using Deep Reinforcement Learning. *arXiv preprint arXiv:2504.21071*.

Zhao, H., Ma, Z., Liu, L., Wang, Y., Zhang, Z., & Liu, H. (2025). Optimized path planning for logistics robots using ant colony algorithm under multiple constraints. *arXiv preprint arXiv:2504.05339*.

Towards a Bayesian Cognitive Model of Self-Belief Updating

Nils Wendel Heinrich[1,2,*], **Rebecca von Engelhardt**[1,*], **Annika Österdiekhoff**[3], **Stefan Kopp**[3], **Nele Russwinkel**[1]
nils.heinrich@uni-luebeck.de
[1]Institute of Information Systems (IFIS), Department of Computer Science and Engineering, Universität zu Lübeck, Lübeck, Germany
[2]Cognitive Psychology and Ergonomics, Technische Universität Berlin, Berlin, Germany
[3]Social Cognitive Systems, Faculty of Technology, CITEC, Bielefeld University, Bielefeld, Germany
[*]These authors contributed equally to this work

Abstract

Bayesian inference is a powerful mathematical framework for modeling cognitive processes, and it has been widely applied in computational models of sensorimotor integration. However, higher-level cognitive functions, such as updating beliefs about one's own sense of control, may also rely on processes that resemble Bayesian integration. We conducted an experiment in which participants navigated a spaceship through sections characterized by temporary control loss. They had to use their knowledge about the environment to anticipate the ship's trajectory several steps ahead. Before each trial, participants rated how well they expected to perform; afterward, they rated how much control they felt they had. Through model selection, we found that both expectancy about one's own performance and objective measures of performance influenced participants' sense of control ratings. Additionally, the control rating from the previous trial predicted expectancy in the following trial, suggesting a sequential updating process. We implemented a sequential Bayesian updating model that integrates prior expectancy and new performance evidence to infer participants' control judgments. We discuss the model's performance and limitations in light of our empirical findings. Our goal is to integrate this framework into a cognitive model using the ACT-R architecture. This is ongoing work, and we share initial results.

Keywords: Action Control; ACT-R cognitive architecture; Bayesian modeling; Belief updating; Cognitive modeling

Introduction

Humans hold beliefs about their ability to act, that are shaped by prior and recent performance and individual weighting of new performance evidence. (Tuckman & Sexton, 1990). A reliable way to model this belief updating over time is Bayesian updating (Moutoussis, Fearon, El-Deredy, Dolan, & Friston, 2014), a framework that has been successfully applied to understand how the brain integrates information from various sources.

Behavior is inherently goal-directed: when a goal is achieved, the behavior is considered effective and associated with good performance; when a goal is not met, the behavior is seen as ineffective and consequently associated with poor performance. Humans generate behavior based on a mental model of the world, an internal representation of their environment. This model helps them interpret objects and dynamics around them, anticipate future states, and make decisions. By relying on this representation, individuals can derive goals they want to achieve and determine the actions needed to bring about the desired changes.

Self-belief about performance, or how capable an individual perceives itself to be, is a part of this mental representation. But the representation and self-belief are also interconnected in a bidirectional way: the representation guides behavior, which in turn affects self-belief, and this updated self-belief as part of the representation influences future behavior. This dynamical system of self-belief and behavior results in very efficient action control when we consider human behavior.

We aim to integrate the Bayesian framework for self-belief updating with a cognitive model of navigation. Our goal is to develop a model that explains behavior as arising from a mental representation of the environment, which includes the agent's self-belief. For this we use our custom-developed experimental environment, in which an agent steers a spaceship while trying to avoid crashing into obstacles. We explore how the mental model of the agent influences self-belief about performance and interacts with personality traits. To achieve this, we analyze and discuss how human experimental data fits a Bayesian updating process. In a next step, we introduce a cognitive model that we implemented in the ACT-R architecture. The ACT-R model features a situated state chunk that reflects the representation of the environment. Behavior is directly produced on the basis of the values held in the chunk slots. We would like to integrate these frameworks to give the model a sense of self-belief. This is ongoing work, and we welcome discussions on how to best combine both approaches.

Action Control

Action control is a broad concept describing how humans interact with their environment. It includes the preparation, execution, and regulation of actions. A core assumption is that human behavior is goal-directed, meaning that actions are executed with a desired state of the environment (or the agent within it) in mind (Frings et al., 2020). To do this, humans rely on a mental representation of the environment, allowing them to anticipate goal states and plan actions accordingly.

In real-world behavior, actions are rarely isolated. Instead, they form sequences of consecutive actions that are closely interconnected. This makes studying action control challenging. Despite these complexities, we focus on shedding light on the process of action preparation. Specifically, we investigate how people form self-beliefs about their own

performance or control that is based on action control. We consider the self-belief to be part of their mental representation of the environment, or more precisely, of themselves within the environment. We propose that action preparation is largely driven by self-belief and that actual performance, in turn, feeds back into and updates this belief, creating a bi-directional relationship.

To model this evolving process of self-belief updating across trials, we plan to apply the Bayesian framework. But first, we examine how inter-individual differences, measured through the Big Five personality traits using the NEO-Five-Factor Inventory (NEO-FFI) (Borkenau & Ostendorf, 2008), are associated with this potential updating process. We present a first iteration of a cognitive model of action preparation, specifying the cognitive processes necessary for effective action planning within a custom-developed experimental environment. This model is implemented using the ACT-R cognitive architecture (Anderson et al., 2004). Ultimately, we aim on combining the Bayesian updating process and the ACT-R model of action preparation. With this, we want to contribute to the discussion on integrating mathematical models of cognition, such as Bayesian frameworks, with cognitive architectures that take a more symbolic and structured approach to modeling cognition.

Before an action is initiated, it must be prepared. A key part of action preparation is the process of selecting an action (Press, Gherri, Heyes, & Eimer, 2010). Often, multiple actions can achieve the same outcome or slightly different ones that still satisfy the goal state, and a choice must be made between the different options. However, humans are not optimal, but rather rational action controllers: we do not always choose the best possible option, but instead exhibit specific biases. Biases become especially pronounced under time constraints, which is common in everyday life. In these situations, there is insufficient time to weigh all possible actions. Affective beliefs, including the self-belief about one's own performance, are always present and influence action control. These beliefs, along with situational information, contribute to the action selection process through top-down processing. In this case, top-down processes help prioritize actions based on past experiences and internal goals.

Once an action is prepared, it is initiated and continuously adjusted based on sensory feedback (Synofzik, Vosgerau, & Newen, 2008). Prediction errors, differences between expected and actual sensory input while the action is executed, guide these adjustments in real time. The ability to correct the course of an action on the fly reduces the demand on the initial preparation/selection process. Action execution with the prediction and matching of sensory feedback, can be modeled using predictive coding (Rao & Ballard, 1999). One key aspect of this framework is that larger prediction errors require larger adjustments to maintain the intended course of action.

Once an action is completed, its outcome can be evaluated: whether or not the action goal was met. This evaluation may trigger adjustments, but unlike the real-time adjustments at the sensorimotor level that refine the course of an action, these occur at a higher cognitive level within the action control hierarchy. If the intended outcome is not achieved, for example, the mental model of the environment may need to be adjusted, as the failure implies that the current representation was inaccurate.

There are two different adjustments that can be made at this level. First, the representation of the environment may be refined, improving how the individual perceives and anticipates interactions within it. Second, the self-belief - a meta-cognitive construct reflecting, among other aspects, how the individual perceives itself as an efficient agent within the environment - may be updated, either strengthened or weakened depending on whether the action goal was achieved or not (Müller-Pinzler et al., 2022). These adaptations influence future action preparation, changing action selection in a way that enhances the likelihood of successfully achieving action goals; interacting more efficiently with the environment.

Self-Belief and Self-Belief Updating

Self-belief is shaped by both stable traits and dynamic, context-dependent confidence. This distinction was shown in fMRI studies, where global self-performance, a trait-like aspect of self-belief, was associated with activity in the ventral striatum, while local confidence, which varies with situational demands, was reflected in frontoparietal network activity (Rouault & Fleming, 2020).

Moreover, evidence suggests that self-belief is continuously updated through Bayesian inference, where prior experiences of meeting action goals are integrated with new evidence about the own performance (Moutoussis et al., 2014). The stable trait of global self-performance influences this updating process, regulating how much weight is given to new evidence that is either positive or negative.

In the following, we present an experiment designed to assess self-belief based on individual performance, with the goal of modeling it as a Bayesian updating process. We also examine all five personality traits of the Big Five, using the NEO-FFI questionnaire, under the assumption that these traits might influence how new evidence is weighted in the self-belief Bayesian updating process.

For the time being, we focus on deriving self-belief from action preparation (based on mental representations) and not from the ability to correct poorly prepared actions during execution. Therefore, we have designed our experimental paradigm in such a way that excludes the possibility to correct actions in real time. This presented a methodological challenge, which we address with an environmental feature we call drift (see experiment section).

Methods

Experiment

The main part of the experiment consists of a game in which participants have to steer a spaceship through a closed environment which is bounded by walls on both sides, left

and right. The experimental environment is implemented in Python (Van Rossum, Guido & Drake, 2009) using the PyGame package (Shinners, 2011) and runs at 60 FPS[1]. The spaceship automatically moves downwards through the environment at constant speed. Participants are tasked to avoid crashing into walls or obstacles and make it to the finish line at the bottom of the environment. They can use the keyboard to move horizontally: [Y] for left and [M] for right. The spaceship is always displayed in the screen center and the environment will shift around the spaceship. Therefore, if participants want to move to the right, the whole environment is shifted to the left instead.

Participants encounter so-called *drift* sections, indicated by a red bar on either side. Once the spaceship enters a drift section with the red bar displayed for example on the right side, the spaceship will be pushed to the left for the duration of the drift section; the length of the red bar (if in contrast the red bar is displayed on the left, the spaceship is pushed to the right). An example of such a drift section can be seen in Figure 1. Within a drift section, it is not possible for the participants to steer, i.e. pressing the Y and M keys does not have any impact on the spaceship's movements. However, the horizontal push of the drift is constant and can therefore be anticipated once the participants have learned about it.

Drift is the key environmental feature that participants must mentally represent to accurately predict the spaceship's trajectory and to effectively prepare for drift sections.

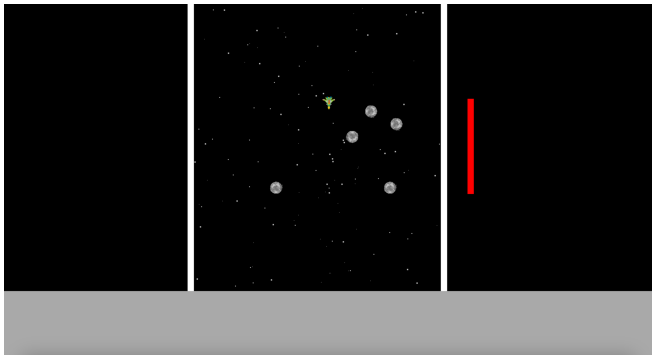


Figure 1: Depiction of an instance within the experimental environment. Shown here is the space of the environment that is drawn on the screen during the experiment. Obstacles are scattered between the two walls on the left and right. The red bar indicates a drift section with push to the left.

The study is divided into multiple levels. Every level consists of four sequential drift sections. Obstacles that need to be avoided are placed solely within drift sections. All drift sections have the same length, but the distance between two following sections and the side on which the red bar is displayed (its push direction) vary. Levels were created randomly including the position of the obstacles, but it was ensured that there is always a possible way for the spaceship to navigate through them.

If participants manage to pass all four drift sections, they reach a green line at the end which means the level is successfully completed. It took on average 18.64 seconds (3.53) to complete a level (deviations may occur due to rendering of the game environment). If the spaceship crashes into an obstacle or in one of the walls, the level ends prematurely and a display is shown with a notification about the crash.

Procedure

Before participants started with the actual trials of the experiment, a training level was presented to familiarize them with the environment and steering the spaceship. The training level consisted of multiple drift sections, first without and then with obstacles. If participants crashed during the training, the training level was presented again until they successfully navigated through it once to ensure the task was sufficiently understood. After that, the trials started. In total, there were 80 different levels. If a participant failed to complete a level, the same level was presented again - up to three times in total - later during the experiment (level was inserted at a random position within the sequence). Before every trial, participants were asked to rate the expectation of their own performance in the upcoming trial ("In the upcoming run, how good do you think you will perform?") on a scale ranging from 1 ("very poorly") to 7 ("excellent") which we will refer to with the term *expectancy* from now on. After each trial, they had to rate their *sense of control* (SoC; "In the most recent run, how strong was your feeling of control?") on the same scale (1-7) which is here used to assess self-belief. Depending on the participants' performance, the main part of the experiment took approximately 40-55 minutes. After having completed the trials, participants were asked to fill out two questionnaires. The first one was the German version of the 30 items NEO-FFI which is commonly used to assess the Big Five personality traits: Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. The second questionnaire was a short four item questionnaire asking for video game expertise such as the frequency and types of games played (Nolan, 2022). From briefing to debriefing and including time for potential questions by participants, the whole experiment took approximately one hour and 10 minutes.

Data Analysis

A total of 26 participants were collected between 09.02.2025 and 12.05.2025 and data collection was completed since submitting the first version of this paper. Below we describe the final model selection.

We used linear mixed modeling (Bates, 2015) in the R programming language (R Core Team, 2022) to analyze the factors influencing responses to both the SoC and expectancy question separately. A Box-Cox distributional analysis (Box & Cox, 1964) showed that the predicted variable SoC required a square root transformation. No transformation was required for expectancy. We included participant ID as a random intercept for both model selections.

[1] Git repository containing the experimental environment.

Either model selection was done applying the following procedure. We determined the fixed effects structure by comparing Bayesian Information Criterion (BIC) values (Chakrabarti & Ghosh, 2011). Since BIC penalizes models with more parameters while also accounting for the number of observations, this approach helps prevent overfitting and ensures a simpler final model. Starting with a full model that included all possible fixed effects, we removed individual variables step by step. After each removal, we compared the new model's BIC with the previous one. Once the fixed effects structure had been determined, we evaluated the random slopes by also referring to the BIC - starting with the full random effects structure and removing individual random slopes step by step. This way, we successively approached the best model structure for the present data set.

All models were fitted using the maximum likelihood criterion and we used a random seed (36) for reproducibility of our results[2].

Results

On average participants successfully solved 24.02% (0.07) of the trials.

For the effects in the linear models predicting SoC and expectancy, we report parameter estimates along with their standard deviations, confidence intervals (obtained from parametric bootstraps with 10,000 iterations), and p-values.

SoC. We predicted SoC responses on a trial-by-trial basis.

Referring to the intraclass correlation coefficent of a null model, participant ID accounted for 46.05% of the total variance in SoC responses, indicating a notable individual difference component.

The initial model included all five dimensions of the NEO-FFI. However, none of these factors remained in the final model after model selection. The final model featured the following predictors: expectancy (response to the expectancy question before the trial), N_{drift} (how far participants progressed in the trial, ranging from 0 to 4), and *crashed* (whether the participant crashed in the current trial).

The higher participants expected their performance to be, the higher they rated their own control during the trial (β = 0.10, σ = 0.01, 95% CI [0.07, 0.12], $p < .01$). Actual performance also influenced SoC. It increased with the number of successfully completed drift sections (N_{drift}; β = 0.07, $\sigma < 0.01$, 95% CI [0.06, 0.08], $p < .01$). Conversely, SoC decreased when participants crashed in the current trial (β = -0.32, σ = 0.04, 95% CI [-0.39, -0.24], $p < .01$).

Expectancy. We again predicted expectancy on a trial-by-trial basis.

In a null model, the intraclass correlation coefficent indicated that participant ID accounted for 58.90% of the total variance in expectancy responses, highlighting significant individual differences also in performance expectations.

[2]Git repository containing data and R analysis script (and also the sequential Bayes model described later).

The final model predicting expectancy included two predictors: $N_{crash-success}$ (the number of times a participant crashed but successfully completed the very next trial) and SoC_{n-1} (the SoC response from the previous trial).

Interestingly, after random slopes had been selected, $N_{crash-success}$ did not yield significance ($\beta < 0.01$, $\sigma < 0.01$, 95% CI [<-0.01, 0.02], p = .153). However, higher SoC ratings during the last trial were associated with significantly greater expectancy, increasing with each point on the 1–7 scale (β = 0.34, $\sigma < 0.01$, 95% CI [0.33, 0.36], $p < .01$).

Discussion of Experimental Data

SoC as self-belief was influenced by both prior expectancy and new evidence about performance. However, against our hypotheses, there were no significant effects for any of the NEO-FFI personality dimensions. We assume that SoC results from a sequential updating process that can be modeled in Bayesian terms, where expectancy serves as the prior and new evidence about performance acts as the likelihood. In this framework, we had expected personality traits to modulate how prior and likelihood are weighted during updating, but the lack of significant personality effects implies a more uniform updating process across individuals.

Expectancy was mainly explained by the last trial's SoC, suggesting that the posterior belief from the previous trial carries over into the next one as prior knowledge, consistent with our Bayesian interpretation. Based on these findings, we have now constructed a mechanistic Bayesian updating model of self-belief that defines the sequential updating process we assume to be underlying SoC.

Sequential Bayesian Updating of Self-Belief

We translated the findings of our model selection directly into the structure of a Bayesian updating model of self-belief. We introduce weighting parameters driving the relative influence of prior expectancy and new performance-related evidence. The prior weight is fixed at 1, a reference, while the weight on the likelihood is treated as a free parameter that the model estimates from the data. This parameterization allows us to examine how strongly participants rely on new evidence relative to their prior expectations when forming SoC judgments.

$\beta_{i,1}$ and $\beta_{i,2}$ denote the participant-specific regression coefficients. The likelihood components are $N_{i,t}^{\text{drift}}$, representing the numerical *drift* evidence on trial t for participant i, and $C_{i,t}$, representing the *crashed* evidence. The log-likelihood weight is then given by:

$$\log w_{i,t} = \beta_{i,1} \cdot N_{i,t}^{\text{drift}} + \beta_{i,2} \cdot C_{i,t} \tag{1}$$

The resulting likelihood weight is:

$$w_{i,t} = \exp(\log w_{i,t}) \tag{2}$$

The predicted SoC, denoted $\mu_{i,t}^{\text{SoC}}$, is computed as a precision-weighted average of prior and likelihood:

$$\mu_{i,t}^{\text{SoC}} = \frac{Prior_{i,t} + w_{i,t} \cdot N_{i,t}^{\text{drift}}}{1 + w_{i,t}} \tag{3}$$

Finally, the observed SoC judgments are normally distributed around the posterior mean:

$$\text{SoC}_{i,t} \sim \mathcal{N}\left(\mu_{i,t}^{\text{SoC}}, \sigma^2\right) \tag{4}$$

Dynamic Likelihood Weighting

In this paper, we cannot examine every participant, but we highlight a small selection as examples. Their individual likelihood weighting results are shown in Figure 2. We divided these participants into low and high performers, based on the idea that performance (i.e., the likelihood) would influence how strongly it is weighted when rating one's own SoC. However, we do not observe a clear difference in likelihood weighting based on performance level. While we expected high performers to give more weight to performance evidence (a), we also see that some high-performing individuals assign relatively low weight to it (though the likelihood weight is still higher than that of the prior which is indicated by the dashed line). Similarly, low-performing individuals (b) may also place a strong emphasis on performance evidence when forming their SoC. We had expected that, especially among low performers, personality traits might help explain these differences in weighting. However, since the personality effects were not significant, we cannot draw firm conclusions about the source of these individual differences.

To evaluate the overall fit of our Bayesian model and its ability to predict individual SoC ratings, we computed the root mean squared error (RMSE) between predicted and observed ratings. The model yielded an RMSE of 0.87 (SD = 0.22), indicating that, on average, predicted ratings deviate from actual ratings by nearly one point on the 7-point SoC scale. Moreover, the model's goodness of fit for individual participants is not explained by their overall success rate. That is, the model does not systematically predict SoC ratings more accurately for high performers than for low performers. While the RMSE shows that the model captures overall patterns in the data, it fails to account for more dynamic within-participant changes in weighting across trials.

Figure 2 suggests that the model underestimates abrupt shifts in how participants rely on performance evidence. For instance, in some cases, a crash followed by a successful trial seems to trigger a sharp increase in the weighting of the likelihood - an effect the model does not fully anticipate and thus may lead to systematic prediction errors in these situations. This is most likely due to the current model's estimated weighting parameter being based only on how many drift sections had been encountered ($N_{i,t}^{\text{drift}}$) and whether there was a crash or not ($C_{i,t}$; Equation 1). These limitations highlight areas where the model could be refined and should be kept in mind when evaluating its predictive performance.

During model selection, we considered $C_{trial-1}$ as a potential fixed effect for SoC, but it was excluded from the final model. One possibility is that the model is currently missing a relevant interaction between $C_{trial-1}$ and N_{drift}, which may be necessary to improve predictive accuracy. A clear next step in our analysis is to revisit the model selection process with this interaction term included.

Once the Bayesian model reliably captures how participants form judgments about their sense of control (SoC) and the information they rely on, we can embed this mechanism into our cognitive model of action preparation. The aim is for the cognitive model to use the current SoC as a basis for action planning and to adapt its behavior dynamically, depending on how much in control it perceives itself to be.

We have already begun developing a model of action preparation that starts with a rough idea of drift (how long it lasts and how far it pushes) and refines its internal representation over time through acting within the DodgeAsteroids environment. In the following section, we provide a conceptual overview of this model.

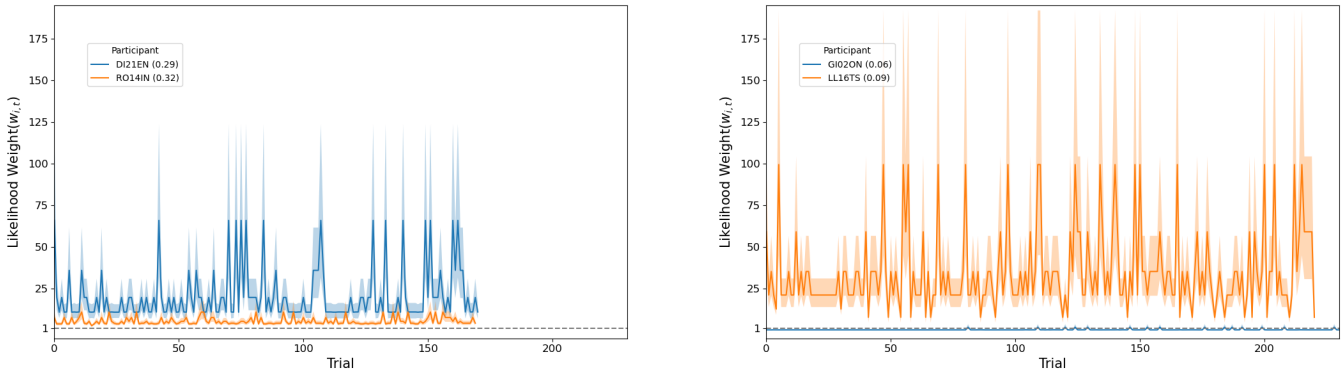
A Computational Cognitive Model of Action Preparation

The ACT-R cognitive architecture (Anderson et al., 2004) separates declarative memory (facts, concepts) from procedural memory (how-to knowledge/rules). This makes it well-suited for our purposes, as the model can encode an initial impression of drift as a chunk in declarative memory, which is then turned into effective procedural strategies. But, using ACT-R for this kind of learning process also presents challenges. It requires careful design of how declarative knowledge, such as different representations of drift, is encoded and retrieved. The model must also handle increased complexity when the environment contains subtle or probabilistic cues, which can be difficult to represent and act on within a symbolic framework. Since the ACT-R model is not the primary focus of this paper, we provide only a brief overview.

Model Components and Functions

Our model of action preparation is implemented using a hybrid approach. Some of the core components are written in Python and executed within ACT-R running in Lisp, using the py4cl library for interoperability. The Python experimental environment and the ACT-R model communicate through ACT-R's RPC interface, managed by a socket connection.

Drift is encoded as a vector in the `situated-state` chunk within declarative memory. This chunk includes slot values for the predicted horizontal displacement (`drift-x`), the estimated duration of the drift (`drift-y`), and the direction from which the drift originates (`drift-direction`), based on the side where the red drift-indicating bar is perceived. In our current model, `drift-x` is initialized with a random value and `drift-y` is set to the true length of the drift section. The chunk also stores the spaceship's current horizontal and vertical position (`agent-x` and `agent-y`), providing a unified representation of both, the environmental dynamics and the agent's state. The chunk is retrieved when the model per-



(a) High performers

(b) Low performers

Figure 2: Dynamic likelihood weighting across trials for four participants, grouped by (a) high and (b) low performance. Axes are scaled equally for better comparison. Dashed lines indicate the fixed prior weight of 1. The low-performing individuals required more trials to complete the experiment and are identified by lower success rates (in parentheses after participant IDs). There is no clear difference in likelihood weighting between high and low performers. In both groups, there are individuals who place considerably more weight on the likelihood, as well as those who weight prior and likelihood nearly equally.

ceives a red object just outside the environment's walls, either on the left or right, which serves as a retrieval cue.

To prepare an action, the model uses the values in the chunk to project a drift vector onto the upcoming drift section, ensuring that the vector does not pass through any of the obstacles. The drift vectors x-component corresponds to `drift-x`, while the y-component corresponds to `drift-y`. The sign of the x-component is determined by `drift-direction` (that is, whether the drift originates from the left or right side of the agent) so that the vector points in the correct horizontal direction when projected. The vector is anchored at the vertical start of the drift section. The model then selects a high-level goal: the position it aims to steer toward, based on the starting point of the projected trajectory. It must learn that not all positions can be reached from the current position of the spaceship, `agent-x` and `agent-y`. Depending on the spaceship's horizontal position, the vertical distance to the drift section might not allow enough time to steer horizontally to the target position given by `high-level goal-x`. This requires balancing between the safest available path and one that is physically achievable.

Motor control in the model is straightforward. Once a high-level goal is selected, the model compares the goal's x-coordinate with the current horizontal position of the spaceship (`agent-x`). If the goal lies to the left, it initiates a Y-key press; if to the right, it initiates an M-key press. In cases where no high-level goal is defined, the model defaults to steering toward the horizontal center of the environment.

It is important to note that we consider this vector-based computation a hack rather than an exact cognitive process. Humans likely do not engage in explicit vector calculus to plan movement. Instead, we assume they rely on heuristics based on spatial features. To better reflect this in upcoming iterations, we will define ACT-R production rules that capture rule-based action preparation.

Next Steps

Our next steps focus on two main objectives: reconsidering the parameterization of the Bayesian updating process underlying self-belief, and refining the computational cognitive model of action preparation. Through the exploration of additional interactions, for example between $C_{trial-1}$ and N_{drift}, we aim to capture more accurately how prior experiences and new performance outcomes dynamically influence belief updating. Additionally, we plan to expand the Bayesian inference mechanism to support probabilistic updating of hypotheses about the drift environmental feature, enabling the model to adapt more flexibly to uncertainty in task dynamics.

We will continue refining our computational cognitive model to align more closely with human behavior, improving its mechanisms for action selection and navigation. The model is supposed to successfully complete trials at a level comparable to human participants. Once this is achieved, we plan to integrate the Bayesian updating framework into the cognitive model, allowing it to explain both performance and self-belief updating in response to task outcomes.

A key challenge remains in effectively combining these two modeling frameworks. One potential method is embedding the mathematical model identified through model selection into the ACT-R cognitive architecture. This would allow the cognitive model to not only replicate human performance but also dynamically update self-belief based on task outcomes. However, we seek to explore alternative ways of integrating these frameworks beyond simply embedding a data-driven mathematical model within ACT-R. Future work will focus on different integration approaches to develop a unified framework that captures both action control and self-belief updating in a cognitively plausible manner.

Acknowledgments

This research was funded by the German Research Foundation (DFG) Priority Program 2134 'The Active Self'.

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.

Bates, D. (2015). *MixedModels: A Julia Package for Fitting Statistical Mixed-Effects Model.* Retrieved from https://github.com/dmbates/MixedModels.jl

Borkenau, P., & Ostendorf, F. (2008). *NEO-FFI : NEO-Fünf-Faktoren-Inventar nach Costa und McCrae, Manual*. Hogrefe. Retrieved from https://pub.uni-bielefeld.de/record/1900471

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *26*(2), 211–243. Retrieved 2024-01-03, from https://academic.oup.com/jrsssb/article-abstract/26/2/211/7028064 (Publisher: Oxford University Press)

Chakrabarti, A., & Ghosh, J. K. (2011, January). AIC, BIC and Recent Advances in Model Selection. In P. S. Bandyopadhyay & M. R. Forster (Eds.), *Philosophy of Statistics* (Vol. 7, pp. 583–605). Amsterdam: North-Holland. Retrieved 2024-01-03, from https://www.sciencedirect.com/science/article/pii/B9780444518620500186 doi: 10.1016/B978-0-444-51862-0.50018-6

Frings, C., Hommel, B., Koch, I., Rothermund, K., Dignath, D., Giesen, C., ... others (2020). Binding and retrieval in action control (brac). *Trends in Cognitive Sciences*, *24*(5), 375–387.

Moutoussis, M., Fearon, P., El-Deredy, W., Dolan, R. J., & Friston, K. J. (2014). Bayesian inferences about the self (and others): A review. *Consciousness and cognition*, *25*, 67–76.

Müller-Pinzler, L., Czekalla, N., Mayer, A. V., Schröder, A., Stolz, D. S., Paulus, F. M., & Krach, S. (2022). Neurocomputational mechanisms of affected beliefs. *Communications biology*, *5*(1), 1241.

Nolan, D. (2022). *Video survey.* https://www.stat.berkeley.edu/~nolan/surveys/videoSurvey.pdf. Retrieved from https://www.stat.berkeley.edu/~nolan/surveys/videoSurvey.pdf (Accessed: 2022-12-08)

Press, C., Gherri, E., Heyes, C., & Eimer, M. (2010). Action preparation helps and hinders perception of action. *Journal of Cognitive Neuroscience*, *22*(10), 2198–2211.

R Core Team. (2022). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from https://www.R-project.org/ (Version 4.1.0)

Rao, R. P., & Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, *2*(1), 79–87.

Rouault, M., & Fleming, S. M. (2020). Formation of global self-beliefs in the human brain. *Proceedings of the National Academy of Sciences*, *117*(44), 27268–27276.

Shinners, P. (2011). *PyGame.* Retrieved 2022-08-29, from http://pygame.org

Synofzik, M., Vosgerau, G., & Newen, A. (2008, March). Beyond the comparator model: A multifactorial two-step account of agency. *Consciousness and Cognition*, *17*(1), 219–239. Retrieved 2023-11-27, from https://www.sciencedirect.com/science/article/pii/S1053810007000268 doi: 10.1016/j.concog.2007.03.010

Tuckman, B. W., & Sexton, T. L. (1990). The relation between self-beliefs and self-regulated performance. *Journal of Social Behavior and Personality*, *5*(5), 365.

Van Rossum, Guido, & Drake, F. L. (2009). *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace.

Using Cognitive Models to Test Hypotheses for a Misinformation-related Effect

[1]Alexander R. Hough (alexander.hough.1@us.af.mil)
[2]Othalia Larue (othalia.larue@parallaxresearch.org)

[1]Air Force Research Laboratory, Wright-Patterson AFB
[2]Parallax Advanced Research, Beavercreek, OH 45431 USA

Abstract

Human's systematic cognitive processes are vulnerable to misinformation-related effects. For example, misleading information can have a lasting influence even after it has been corrected. This continued influence effect (CIE) was found to be resistant to mitigation in experiments. Leading explanations are memory-based, but some include emotion and/or reasoning. However, mixed findings have hindered our understanding of the phenomenon. We argue that cognitive models are uniquely suited to help clarify these mixed findings and theories through specification of underlying mechanisms, testing hypotheses, and identifying why and when mitigations are effective. We start by discussing relevant experimental findings, then present an updated cognitive model of the CIE, compare model fits to two experiments across three model variations, and discuss the results along with recent exploratory analyses with previous experiments.

Keywords: ACT-R; Cognitive modeling; misinformation; continued influence effect; knowledge representation; affect

Introduction

Humans often leverage heuristics that exploit statistical regularities (i.e., cues) in the environment, which can lead to successes (Gigerenzer & Gaissmaier, 2011) and systematic failures (Kahneman, 2011). Failures are usually related to applying a heuristic in the wrong context or when the environment is "hostile" and the diagnostic cues are misleading, limited, or manipulated (Stanovich, 2018). These heuristic vulnerabilities were successfully leveraged to spread misinformation in the past (Lewandowsky et al., 2013) and can now be spread more quickly (Vosoughi et al., 2018). Here, we focus on a specific vulnerability, the continued influence effect (CIE; (Johnson & Seifert, 1994; Lewandowsky et al., 2012)), where misinformation is presented and has a persistent effect on decisions even after it has been corrected.

The CIE

CIE experiments typically present two narratives about scenarios (i.e., events), where one contains misinformation and the second has a correction. In general, corrections reduce, but do not eliminate misinformation reliance (Ecker & Antonio, 2021; Ecker et al., 2017; Johnson & Seifert, 1994). Explanations suggest memory is permanent, but varies in strength due to re-activation or association with different information (Wilkes & Leatherbarrow, 1988). One account focuses on issues with memory retrieval, such as competing memory activations (Ayers & Reder, 1998; Ecker et al., 2010), recency effects (Ecker et al., 2015), or familiarity-based fluency (Ecker et al., 2011). These processes can lead to errors. However, if detected, errors can be avoided through directed retrievals and/or reasoning, leading to more thorough consideration of information (Pennycook & Rand, 2019; Stanovich, 2018). An alternative account involves the preference for mental models that are more complete or coherent, which may involve causal information, believability, or interconnected details (Gentner, 1983; Johnson & Seifert, 1994; Lewandowsky et al., 2012; Wilkes & Leatherbarrow, 1988). Misinformation may be preferred when it is part of a more complete mental model or the correction cannot be integrated, which may create a feeling of discomfort (Ecker et al., 2011; Susmann & Wegener, 2022). Feelings alone can also influence memories by making them easier to recall (Yonelinas & Ritchey, 2015), with a greater effect for negative information (Vaish et al., 2008; Williamson et al., 2019). However, the extent emotion influences memory related to misinformation is still unclear (Phillips et al., 2024).

Research covered a range of correction methods to mitigate and explain the CIE (Prike & Ecker, 2023; Walter & Tukachinsky, 2020), and some individual differences (e.g., Brydges et al. (2018)), however, findings are rather inconsistent. There are likely interactions between the content of events, sources of influence, individual differences (e.g., prior experience), and mitigations (Hough et al., Under review). We argue computational cognitive models could help provide a clearer understanding of the CIE by testing explanations in different contexts and help to determine how, when, and why context and sources of influence interact. Here, we extend a basic computational cognitive model framework (Hough & Larue, 2024) by including an additional model variant and experimental dataset to show differences resulting from degree of encoding, affect, and memory rumination.

Modeling the CIE

We implemented our CIE model within the ACT-R cognitive architecture (Anderson, 2007). ACT-R is a hybrid cognitive architecture with symbolic and sub-symbolic structures, and modules representing systems of the mind. The CIE model uses the goal module for the model's focus and storage of goal-relevant information. The vision module allows visual perception, and the imaginal module serves as temporary working memory. The declarative memory module stores in-

formation as chunks and captures memory dynamics. The procedural memory module uses condition-action rules (i.e., productions) to represent knowledge about how to do things and to drive behavior. Here we focus on declarative memory and affect mechanisms to capture the CIE and compare three model variants we used to fit data from two experiments. All model materials are accessible at: `http://act-r.psy.cmu.edu/?post_type=publications&p=35275`.

Declarative Memory and Affect Mechanisms

Through chunk activation, ACT-R declarative memory can capture memory retrieval and mental model effects. Chunks are the basic unit of declarative memory. They include slots with values and have an activation value corresponding to the probability and speed of its retrieval in a given situation (Anderson, 2007). Chunks often compete because they match a retrieval request, but the chunk with the highest activation is more likely to be retrieved. Here, we reduced the terms of the standard activation equation to just base level activation, B_i, which represents recency and frequency of chunk use, and activation noise, ε_i, representing variability in memory. The base level term, B_i, is important for opposing dynamics of learning with experience and forgetting across time: n_i, is the number of times chunk i has been used or retrieved, t_{ij} is elapsed time in seconds since the j^{th} retrieval, and $d \in [0,1]$ is a decay parameter. If used, the base-level constant, β_i, is a constant offset to the equation.

$$A_i = B_i + \varepsilon_i + (V_i * vw) + (Ar_i * aw) \quad B_i = \log\left(\sum_{j=1}^{n_i} t_{ij}^{-d}\right) + \beta_i \quad (1)$$

We added valuation, V_i, and arousal, Ar_i, terms to the equation, with their accompanying weights (i.e., *vw* and *aw*). These terms approximate emotion dynamics based on the core affect theory of emotion (Russell & Barrett, 1999), which focuses on feelings underlying emotion using two dimensions: valuation (positive or negative) and arousal (magnitude). Previous work developed a valuation module (Juvina et al., 2018) to compute the valuation, V_i, and arousal, Ar_i, terms. The current valuation of chunk i at the j^{th} use is based on its previous valuation $V_i(j-1)$ and the difference between the previous valuation and current reward $R_i(j)$ multiplied by a valuation learning rate av. Arousal is the absolute magnitude of valuation and represents the importance of a chunk:

$$V_i(j) = V_i(j-1) + \text{av}[R_i(j) - V_i(j-1)] \qquad Ar_i(j) = abs(V_i(j)) \quad (2)$$

Valuation and arousal are updated each time a chunk is referenced within a time window. They affect activations and can be used as retrieval cues. This aligns with research suggesting emotional memories are more accessible (Buchanan, 2007) and/or easier to recall (Yonelinas & Ritchey, 2015). For example, if negative affect was associated with a chunk, its activation would increase. It could persist over time and affect decision-making despite the accumulation of conflicting, more neutral, evidence.

Table 1: Declarative and valuation parameters for the Ecker and Antonio (2021) experiment and Bruns et al. (2023) experiment (in parentheses if different).

	Memory	Memory-affect	Updated-memory
Retrieval threshold	0	0	0
Base-level decay	0.5	0.5	0.5
Activation noise	0.25	0.25	0.45 (0.25)
Base-level constant	2.5	2.5	0
Declarative finst num	1000	1000	1000
Declarative finst span	1000	1000	1000
Initial valuation	-	1	-
Valuation weight	-	2	-
Valuation learning	-	1	-
Valuation time window	-	0.5	-
Arousal weight	-	1	-
Word affect scaling	-	10 (1)	-
Source affect	-	0.4-1.4 (5)	-

Model Descriptions and Processes

In this paper, we compare and contrast three CIE model variants: Memory, Memory-affect, and Updated-memory. Despite having features of both the memory error and mental model explanations above, we did not explicitly create any variant to directly represent them. Variants have features of both, but we argue the Memory and Memory-affect variants better align with the memory encoding/retrieval error explanation, and the Updated-memory with the mental model explanation. The Memory variant was used as a base for the other two model variants. Changes from the base model (i.e., Memory) included parameter values (Table 1), productions (Figure 1), and/or the activation equation (Equation 1). We start by describing the Memory variant and move on to the other two variants.

Memory The Memory variant used six declarative memory parameters and the first three were previously discussed: 1) base-level constant, β_i, was set to 2.5 (0 is default), 2) base-level decay, d was set to the default of .5, 3) activation noise, ε_i, was set to .25 (recommended range is .2-.8), 4) Retrieval threshold restricts which chunks can be retrieved based on activation and was set at the default value of 0. The last two parameters: 5) declarative finsts that sets number of items marked as retrieved, and 6) declarative finst span that sets the

Table 2: Full football scenario (Ecker & Antonio, 2021) with misinformation in bold and correction italicized.

Football Scenario
Stockholm FC star player Emil Larsson will not be available for the opening match of the Swedish Superettan league season.
Larsson is believed to have tested positive to performance enhancing drugs.
The 27 year-old signed with Stockholm at the beginning of the 2012 season and has since become one of their strongest players.
Larsson scored 23 goals in his first season with Stockholm, and gave 11 assists.
Club president Asgeir Soerenssen, who recently refused several lucrative offers to sell Larsson, was not available for comments.
Recent acquisition Lucas Johansson is predicted to take Larsson's position in the opening round match against arch-rival Goteborg SK.
Oliver Lindgren, SOURCE, stated that "I do not believe that Larsson has engaged in drug use."
Under recently introduced rules, players suspended for drug-related offenses will not receive pay throughout the duration of their suspension.

Table 3: Football scenario parsed into word pairs/chunks and categorized as neutral, misinformation, and correction.

Type	Word Pairs/chunks
Neutral	(Stockholm star-player) (star-player Emil-Larsson) (Emil-Larsson not-available) (not-available opening-match) (club-president Asgeir-Soerenssen) (Asgeir-Soerenssen refused-sale) (refused-sale Emil-Larsson) (Stockhold aquired) (aquired Lucas-Johansson) (Lucas-Johansson replace) (replace Asgeir-Soerenssen) (performance-drugs suspended) (suspended no-pay)
Misinformation	(Emil-Larsson tested) (tested positive) (positive performance-drugs)
Correction	(Oliver-Lindgren source) (source statement) (statement I-do-not-believe-that-Larsson-has-engaged-in-drug-use) (Emil-Larsson not-engaged) (not-engaged performance-drugs)

time items remain marked, were both essentially shut off by setting higher than recommended (i.e., 1000) to prevent an endless loop of retrievals.

The Memory variant focused on retrievals and chunk activation changes based on declarative memory dynamics to capture competition between misinformation and corrections. There are two main sets of processes that: 1) read text and create chunks, and 2) navigate chunks in memory by chaining them together based on associations and then giving a simulated summary of the content (Figure 1). Before information was presented to the model, we parsed the experimental materials (Table 2) into word pairs (Table 3). We provided an in-context learning example to ChatGPT (e.g., Romero et al. (2023)) and had to use the output as a guide to manually generate words pairs for most of the materials due to dramatic changes to ChatGPT (we are working on a better long-term solution). During the first set of processes, the model is presented with word pairs in Table 3 one at a time. The model finds, attends, and reads a word. If the model can't associate words (i.e., only one word was read), then searches for another word to read (i.e., keep-reading). Once two words are read, the model attempts to retrieve (i.e., retrieve-assoc) a matching chunk and if not possible, it creates a new chunk (i.e., create-assoc). Once all experimental materials are read and encoded into memory, the model starts the memory chaining processes. It starts by openly retrieving a chunk for a random scenario (i.e., retrieve-scenario-info) and encoding it (i.e., encode-scenario-info). Next, it attempts to find the root or start of chunk chain through back-chaining (i.e., find-chain-root) using the first word of the encoded chunk (e.g., **tested** positive) and trying to retrieve a chunk with a matching second word (e.g., Emil-Larsson **tested**).

If a chunk was retrieved, it is encoded (i.e., encode-back-chain), otherwise the root is considered found and back-chaining stops (i.e., found-root). Next, the model attempts to parallel chain using the first word of the root (e.g., **Emil-Larsson** not-available) to find a chunk with the same first word (e.g., **Emil-Larrson** tested). Parallel chaining prevents some chunks from being neglected and can link information from different chains. If such a chunk is retrieved, it is encoded (i.e., encode-parallel-chain) and if not, forward chaining is started (i.e., start-forward-chain). The model uses

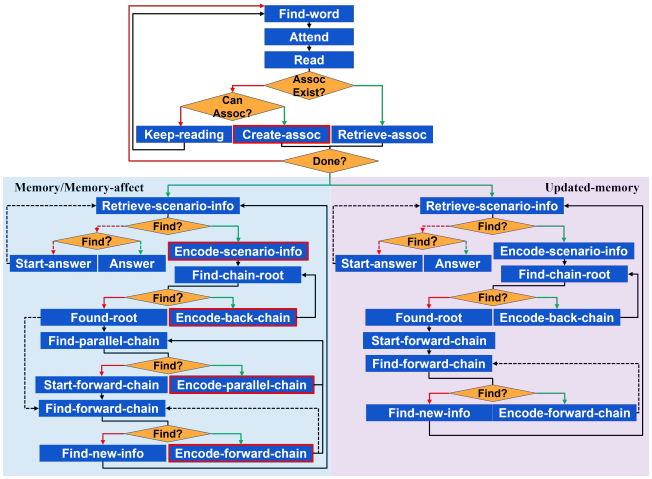


Figure 1: CIE Model Framework Processes.

the second word of the currently encoded chunk (e.g., Emil-Larsson **tested**) to find a chunk with a matching first word (e.g., **tested** positive). If a chunk is retrieved, it is encoded (i.e., encode-forward-chain) and if not, the chain is considered complete and the model starts a new chain (i.e., find-new-info). If the open retrieval (i.e., retrieve-scenario-info) fails because chunks are marked as recently retrieved and/or their activations are below the retrieval threshold, memory navigation is complete and the model prepares a summary answer (i.e., start-answer). The model uses the same navigation processes, excluding parallel chaining (see dotted lines), to find the most active chunk and the chain it belongs to; which is given as the answer (i.e., answer).

Memory-affect The Memory-affect variant included valuation and arousal terms that were added to the activation equation (equation 1), and affect was associated with content when reading and reinforced during retrieval/encoding (red borders in Figure 1). We used valance and arousal values scaled from 0-1 (i.e., below .5 is negative and above is positive) from the 20,000 word NRC Emotion Lexicon Mohammad (2018) to specify affect associated with words. We collapsed valence and arousal values into a single positive reward (i.e., greater than 0) so that negative valence had a greater value, $arousal * (1 - valence)$. We scaled values (Table 1) so that they were high enough to influence chunk activations. We specified source affect based on source credibility and trustworthiness ratings collected in the two experiments we used for model fitting (Bruns et al., 2023; Ecker & Antonio, 2021). These values ranged from .4-1.4 across the five source conditions for one experiment (Ecker & Antonio, 2021) and were set to 5 for a single source for the other (Bruns et al., 2023) Based on previous work with the valuation module (Juvina et al., 2018), we used rewards to update both valuation and arousal terms in the activation equation. These rewards were scaled differently based on valance and arousal values for words (0-1 * scaling value) and ratings for information sources for the two experiments (Table 1). To

align with previous research (Vaish et al., 2008; Williamson et al., 2019), words with negative valence increased activations more than positive. We used five valuation parameters: 1) valuation weight (2), 2) valuation alpha or learning rate (1), 3) valuation time window (.5), 4) arousal weight (1), and 5) initial chunk valuation (1).

Updated-memory We leveraged declarative finsts to restrict retrievals to chunks not recently retrieved for the open retrieval (i.e., retrieve-scenario-info) and during backward, parallel, and forward chunk chaining to prevent infinite chaining loops with additional datasets. The downside was that some chunk chains were unable to complete because some of their chunks had already been retrieved. Therefore, interconnected chunks (i.e., chunks that can chain or link to many other chunks) were not retrieved as much as originally intended and their activation was less affected by memory navigation. We argue that this results in a less coherent mental model. The Updated-memory variant was developed to address this issue. Our initial solution, which still had limitations that we address in the discussion, was to modify the open retrieval (i.e., retrieve-scenario-info) that started the process for each chain and to reset declarative finsts (i.e., chunks marked as retrieved) at the end of each chain. We created another list of non-retrieved chunks so that the model would retrieve each chunk for a given scenario and find the chain it belonged to. The result is that more interconnected chunks (i.e., are present in more chains) are retrieved significantly more. In addition, we eliminated the parallel chaining productions (Figure 1), set the base-level constant, β_i, to the default of 0, and increased activation noise to .45 (Table 1).

Ecker and Antonio (2021) Experiment 1

In experiment 1 from Ecker and Antonio (2021), 53 participants (62% female and mean age 18.6) from the University of Western Australia read narratives about six scenarios (i.e. anti-viral drug, fishing restrictions, food additives, football scandal, joint condition treatments, and water contamination) with embedded misinformation and corrections. Each scenario had a different correction source that varied

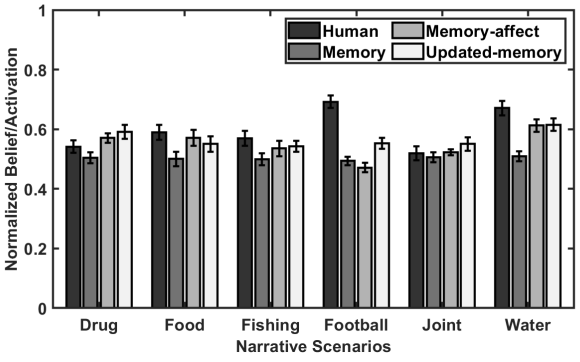


Figure 2: Misinformation belief/activation across scenarios for human and model data. Error bars are SEM.

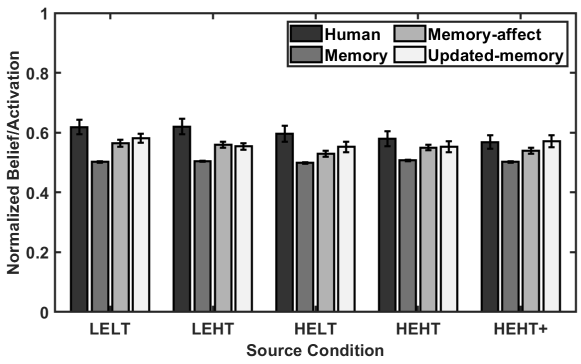


Figure 3: Misinformation belief/activation across source conditions for human and model data. Error bars are SEM

in "quality" (i.e., no retraction [NoR], low expertise/trust [LELT], low expertise/high trust [LEHT], high expertise/low trust [HELT], high expertise/trust [HEHT], and highest expertise/trust [HEHT+]), and was counterbalanced across six versions. Participants answered 10 open-ended and interference questions to comprise a misinformation reliance score, and rated beliefs in misinformation and corrections. Here, we focused on belief scores and approximated them by calculating the average activation for misinformation and correction chunks. We normalized belief scores and activation for misinformation by dividing values for misinformation by the sum of misinformation and correction values. We simulated 60 participants (i.e., 10 for each counterbalancing version) for each model variant. We first compared model fits across narrative scenarios using a correlation and root mean squared error (i.e., *RMSE*), however, we emphasize *RMSE* as the main indicator of fit. The Updated-memory variant had the lowest *RMSE*, $r(8) = 0.11, p = 0.86, RMSE = 0.04$, followed by the Memory-affect variant, $r(8) = 0.65, p = 0.24, RMSE = 0.05$, and the Memory variant, $r(8) = -0.10, p = 0.87, RMSE = 0.10$ (Figure 2).

Next, we compared model variants for source conditions, which were arranged from least to highest quality. We note that only the Memory-affect model is sensitive to the quality of source information. The Updated-memory variant had the lowest *RMSE*, $r(10) = 0.29, p = 0.57, RMSE = 0.07$, followed the the Memory-affect variant, $r(10) = -0.09, p = 0.86, RMSE = 0.10$, and the Memory variant, $r(10) = -0.32, p = 0.53, RMSE = 0.12$ (Figure 3).

Bruns et al. (2023) Experiment

In Bruns et al. (2023), 2,614 participants were recruited from Europe and read one of three claims (i.e., misinformation) about climate change and a separate correction article that came before (i.e., prebunk) or after (i.e., debunk) the claim with or without a source. Similar to experiment 1, sources had associated affect. There was no associated affect for no source conditions. As there were three claims and five conditions (no correction control, prebunk no source, prebunk

source, debunk no source, and debunk source), there were 15 separate conditions. Participants answered questions about agreement with claims (misinformation), credibility of corrections, and intention to engage in discussion. Here, we focus on the agreement with claims, which is comparable to beliefs, and fit the models to reduction in agreement. Reduction in agreement was calculated by subtracting the control group misinformation agreement [activation] from misinformation agreement [activation] for all conditions (i.e., prebunk and debunks with and without source information). We simulated 150 participants (i.e., 10 for each of the 15 conditions) for all model variants. The Updated-memory model has the lowest *RMSE*, $r(6) = 0.93, p = 0.07, RMSE = 0.06$, followed by the Memory-affect variant, $r(6) = 0.70, p = 0.30, RMSE = 0.07$, and the Memory variant, $r(6) = 0.88, p = 0.12, RMSE = 0.24$ (Figure 4).

Discussion

We argued computational cognitive models could help resolve mixed findings and ambiguity with CIE explanations, and compared three CIE model variants. The Memory variant included creation of chunks and limited memory navigation that influenced chunk activation based on retrieval frequency. The Memory-affect variant was an extension that included affect associated with words and information sources at chunk creation, which was reinforced during retrieval. The Updated-memory variant eliminated some processes, better aligned with default parameter values, and increased the extent of memory navigation affecting activation of the more interconnected chunks within narratives.

We argue that the Memory and Memory-affect variants best aligned with the memory encoding/retrieval error explanation and the Updated-memory aligned best with the mental model explanation. Although the Updated-memory model was only slightly better than the Memory-affect variant, it was simpler. These results suggest misinformation beliefs in the two experiments may be best explained by the mental model explanation. These findings also raise a concern for CIE experiments, as some results might be due to the wording of the narrative, which should be considered when testing manipulations to reduce the CIE. However, models developed to directly test specific explanations or hypotheses would provide more clear conclusions. Despite successes, we discuss several limitations that may limit generalization of results.

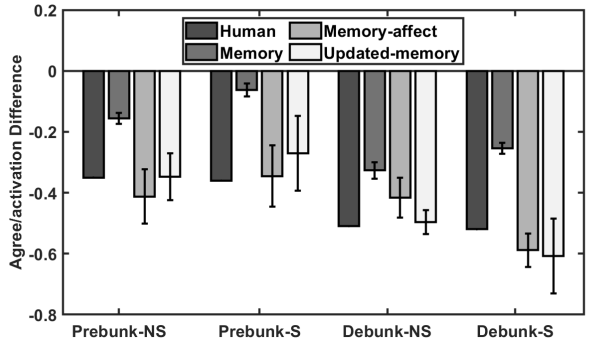


Figure 4: Misinformation agreement/activation difference from control (no correction) across correction type and source conditions for human and model data. Error bars are SEM.

Limitations and Future Work

Our current work has several limitations: 1) We manually generated predicates, 2) we focused on declarative memory with syntactic chunk chaining, 3) we focused on belief and agreement as inferential reasoning was too complex for our current model, 4) we simplified valuation and arousal terms to a single reward value that favored negative affect, and 5) we fit data for only two experiments, and 6) models were developed and compared to theoretical explanations post-hoc.

Misinformation is often in text format, which is a challenge for modeling. Our initial strategy was to parse the scenarios using ChatGPT, but we had to manually generate predicates for most scenarios after ChatGPT was updated. The parsing and subsequent representation of written content has a large impact on a model's behavior, and in our case, the model is dependent on the structure of representations to chain and connect information. We are currently finding a more stable language processing method and are currently improving chaining by: 1) treating words (i.e., rather than word pairs) as chunks, 2) using spreading activation to reduce memory retrievals, and 3) using semantic and affective information rather than word matching to connect information. This will enable comparing spreading activation and memory navigation mechanisms, more holistic knowledge representation, and extend question answering capabilities.

We focused on beliefs and agreement, and approximated them by the average activation of misinformation and correction chunks. Most experiments include several open-ended and inference questions. Our planned improvements to language parsing and knowledge representation will extend the model's behavior capabilities, but we also need to add some reasoning ability and/or causal knowledge.

Our emotion implementation used a previous valuation module to update chunk activation through rewards, which combined valuation and arousal values and favored negative affect. Rather than rewards, we plan to associate valuation and arousal values with words to allow for differential influence of both negative and positive affect.

We fit our model to only two experiments, which limits generalization. Our improvements mentioned above should increase the ability of the model to handle different content and provide different types of responses. In addition, this will enable more direct testing of explanations and hypotheses. Our goal is to capture fundamental cognitive processes and knowledge representations underlying the CIE and related phenomena.

Conclusion

Overall, we were able to simulate the CIE with and without emotion for two datasets. Planned future improvements will enable exploration of social factors, group interactions, and theoretical explanations across experiments and datasets. Our current work, mixed findings, and interactions between sources of influence, content, and mitigations suggest we may benefit from a methodological re-framing. We suggest treating sources of influence and mitigations as cues could enable predictions similar to multiple-cue decision making (Lee et al., 2019; Weber & Johnson, 2009), and some literature has already suggested source information (Traberg et al., 2024) and emotion (Phillips et al., 2024) serve as cues.

Acknowledgments

This research was supported by the U. S. Air Force Research Laboratory's 711^{th} Human Performance Wing, Cognition and Modeling Branch. Contents were reviewed and approved for public release (AFRL-2025-1039). The views expressed are those of the authors and do not reflect the official guidance or position of the United States Government, the U.S. Department of Defense, the U.S. Air Force, or any of their subsidiaries or employees.

References

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

Ayers, M. S., & Reder, L. M. (1998). A theoretical review of the misinformation effect: Predictions from an activation-based memory model. *Psychonomic Bulletin and Review*, *5*, 1–21.

Bruns, H., Dessart, F. J., Krawczyk, M. W., Lewandowsky, S., Pennycook, G., Pantazi, M., ... Smillie, L. (2023). Investigating the role of source and source trust in prebunks and debunks of misinformation in online experiments across four eu countries. *Scientific Reports*, *20723*.

Brydges, C. R., Gignac, G. E., & Ecker, U. K. (2018). Working memory capacity, short-term memory capacity, and the continued influence effect: A latent-variable analysis. *Intelligence*, *69*, 117–122.

Buchanan, T. W. (2007). Retrieval of emotional memories. *Psychological Bulletin*, *133*(5), 761.

Ecker, U. K., & Antonio, L. M. (2021). Can you believe it? An investigation into the impact of retraction source credibility on the continued influence effect. *Memory & Cognition*, *49*, 631–644.

Ecker, U. K., Hogan, J. L., & Lewandowsky, S. (2017). Reminders and repetition of misinformation: Helping or hindering its retraction? *Journal of Applied Research in Memory and Cognition*, *6*(2), 185–192.

Ecker, U. K., Lewandowsky, S., Cheung, C. S., & Maybery, M. T. (2015). He did it! she did it! no, she did not! multiple causal explanations and the continued influence of misinformation. *Journal of Memory and Language*, *85*, 101–115.

Ecker, U. K., Lewandowsky, S., Swire, B., & Chang, D. (2011). Correcting false information in memory: Manipulating the strength of misinformation encoding and its retraction. *Psychonomic Bulletin & Review*, *18*, 570–578.

Ecker, U. K., Lewandowsky, S., & Tang, D. T. (2010). Explicit warnings reduce but do not eliminate the continued influence of misinformation. *Memory & Cognition*, *38*, 1087–1100.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, *7*(2), 155–170.

Gigerenzer, G., & Gaissmaier, W. (2011). Heuristic decision making. *Annual Review of Psychology*, *62*, 451–482.

Hough, A. R., Arakal, A., & Larue, O. (Under review). Scenarios impact the continued influence effect. *Computational and Mathematical Organization Theory*.

Hough, A. R., & Larue, O. (2024). Exploring memory mechanisms underlying the continued influence effect. *In Proceedings of the 22nd International Conference on Cognitive Modeling. Via mathpsych.org/presentation/1605*.

Johnson, H. M., & Seifert, C. M. (1994). Sources of the continued influence effect: When misinformation in memory affects later inferences. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *20*(6), 1420–1436.

Juvina, I., Larue, O., & Hough, A. (2018). Modeling valuation and core affect in a cognitive architecture: The impact of valence and arousal on memory and decision-making. *Cognitive Systems Research*, *48*, 4–24.

Kahneman, D. (2011). *Thinking fast and slow*. New York: Farrar, Straus and Giroux.

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

Lewandowsky, S., Ecker, U. K., Seifert, C. M., Schwarz, N., & Cook, J. (2012). Misinformation and its correction: Continued influence and successful debiasing. *Psychological Science in the Public Interest*, *13*(3), 106–131.

Lewandowsky, S., Stritzke, W. G., Freund, A. M., Oberauer, K., & Krueger, J. I. (2013). Misinformation, disinformation, and violent conflict: From iraq and the "war on terror" to future threats to peace. *American Psychologist*, *68*(7), 487–501.

Mohammad, S. (2018). Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (volume 1: Long papers)*, 174–184.

Pennycook, G., & Rand, D. G. (2019). Lazy, not biased: Susceptibility to partisan fake news is better explained by lack of reasoning than by motivated reasoning. *Cognition*, *188*, 39–50.

Phillips, S., Wang, S. Y. N., Carley, K. M., Rand, D., & Pen-

nycook, G. (2024). Emotional language reduces belief in false claims. *OSF Preprint*.

Prike, T., & Ecker, U. K. (2023). Effective correction of misinformation. *Current Opinion in Psychology*, 101712.

Romero, O. J., Zimmerman, J., Steinfeld, A., & Tomasic, A. (2023). Synergistic integration of large language models and cognitive architectures for robust ai: An exploratory analysis. *Proceedings of the AAAI Symposium Series*, *2*(1), 396–405.

Russell, J. A., & Barrett, L. F. (1999). Core affect, prototypical emotional episodes, and other things called emotion: dissecting the elephant. *Journal of Personality and Social Psychology*, *76*(5), 805.

Stanovich, K. E. (2018). Miserliness in human cognition: The interaction of detection, override and mindware. *Thinking & Reasoning*, *24*(4), 423–444.

Susmann, M. W., & Wegener, D. T. (2022). The role of discomfort in the continued influence effect of misinformation. *Memory & Cognition*, *50*(2), 435–448.

Traberg, C. S., Harjani, T., Roozenbeek, J., & van der Linden, S. (2024). The persuasive effects of social cues and source effects on misinformation susceptibility. *Scientific Reports*, *14*(1), 4205.

Vaish, A., Grossmann, T., & Woodward, A. (2008). Not all emotions are created equal: the negativity bias in social-emotional development. *Psychological Bulletin*, *134*(3), 383.

Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science*, *359*(6380), 1146–1151.

Walter, N., & Tukachinsky, R. (2020). A meta-analytic examination of the continued influence of misinformation in the face of correction: How powerful is it, why does it happen, and how to stop it? *Communication Research*, *47*(2), 155–177.

Weber, E. U., & Johnson, E. J. (2009). Mindful judgment and decision making. *Annual Review of Psychology*, *60*(1), 53–85.

Wilkes, A., & Leatherbarrow, M. (1988). Editing episodic memory following the identification of error. *The Quarterly Journal of Experimental Psychology*, *40*(2), 361–387.

Williamson, J. B., Drago, V., Harciarek, M., Falchook, A. D., Wargovich, B. A., & Heilman, K. M. (2019). Chronological effects of emotional valence on the self-selected retrieval of autobiographical memories. *Cognitive and Behavioral Neurology*, *32*(1), 11–15.

Yonelinas, A. P., & Ritchey, M. (2015). The slow forgetting of emotional episodic memories: an emotional binding account. *Trends in Cognitive Sciences*, *19*(5), 259–267.

Skill Acquisition from a Bottom-Up Perspective

Yang Ji (y.ji@rug.nl)
Jacolien van Rij (j.c.van.rij@rug.nl)
Niels A. Taatgen (n.a.taatgen@rug.nl)
University of Groningen, Nijenborgh 9, 9747 AG Groningen, Netherlands

Abstract

The PRIMs cognitive architecture utilizes a bottom-up approach, replacing fixed processing sequences with general-purpose operators that dynamically form task-specific procedures. However, it currently predicts only immediate operators contextually, which limits its ability to model goal-oriented, multi-skill task performance. To address this, we introduce a skill buffer containing skill chunks. These chunks enable procedure differentiation by providing: (a) expected outcome operators (stop operators), and (b) context-operator associations that guide operator sequences. Skills can also be inferred from preceding skills and model contexts, potentially facilitating the acquisition of multi-skill processing sequences for more complex tasks. This skill implementation thus provides a bottom-up framework connecting levels of abstraction from operators to skills and multi-skill task performance.

Keywords: contextual learning; procedural learning; skill acquisition

Introduction

A key challenge in cognitive science is to design a unified framework for explaining a diverse set of cognitive tasks. Such a framework is called a *cognitive architecture* – it is both a theory of cognition and a framework for implementing computational simulations of the processes involved in performing cognitive tasks (called *cognitive models*). Notably, even before the development of the very first cognitive architecture, Newell, Shaw, and Simon (1958) had set forth a clear vision: such an architecture, they posited, should demonstrate "specifically and in detail how the processes that occur in human problem solving can be compounded out of elemental information processes" and "how sequences of simple processes could account for the successful solution of complex problems."

Nonetheless, conventional cognitive architectures such as ACT-R and SOAR (Anderson, 2007; Laird, Lebiere, & Rosenbloom, 2017) continue to depend on rigid, preprogrammed processing steps triggered by predefined conditions. This reintroduces the original inquiry of how these processing steps or procedures are learned. A relatively new cognitive architecture known as *prim*itive information processing element*s* (PRIMs) theory decomposes complex production rules into simpler, more flexible building blocks called *primitive operators*. This novel, bottom-up approach enables learning different procedures through a learning mechanism (Taatgen, 2013).

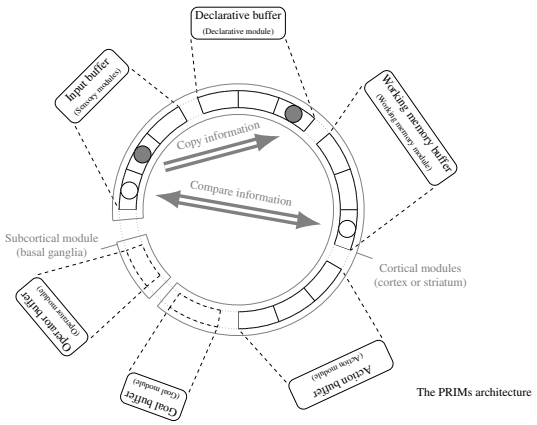


Figure 1: The PRIMs architecture follows a conventional cognitive architecture design, consisting of cortical modules and a subcortical module. Within this structure, primitive operators perform content encoding (single arrow) and comparison (double arrow).

We have previously demonstrated that PRIMs enables general-purpose operators to become context-dependent through contextual learning, forming operator sequences or procedures (Ji, van Rij, & Taatgen, under revision). However, a gap exists between the capabilities of this procedure-based approach and the goal-oriented skills required for typical cognitive task performance. In this technical paper, we will briefly review standard PRIMs' use of contextual learning to reframe and unify traditional rule-based cognitive architecture models. We will then introduce skill-based extensions to this standard approach, and describe the implementation of these extensions.

Reframing Rule-Based Models

A cognitive architecture is generally structured as a series of modules (corresponding to different cortical areas) that communicate through connections via a global workspace (corresponding to the basal ganglia/striatum). These modules are mapped to *buffers* for storing and exchanging information. Operators are the basic link between two module buffers within the workspace. An *operator*, for instance, can encode the content of one buffer into another, or compare whether

the contents of two buffers are identical or different (see Figure 1). Traditionally, these operators were manually composed into rule-like operator sequences (production rule sequences) to explain cognitive processes. These production rules require explicit context-dependent condition checking. In contrast, PRIMs adopts a bottom-up approach. Complex production rules are decomposed into minimal information processing units, or primitive operators. These operators are more context-free and general-purpose than production rules. For a novel task and its immediate context (i.e., the content within the buffers), the model examines the applicability of operators through trial and error. If applicable, the operator's processing is carried out, updating the model's context. After processing, the operator itself also becomes part of the model context. The model then proceeds to select a next operator for the updated context, and this process iterates until terminated by an operator that recognizes the completion of the current task (more details later). In standard PRIMs, the sequence of selected operators is referred to as a *procedure*.

A crucial learning mechanism in PRIMs is its *contextual learning*, which learns to associate the model context (buffer contents, and preceding operator if any) with operators, so that it can select an appropriate action once the model recognizes a task pattern. This mechanism replaces explicit conditional checking in conventional cognitive architectures. The learned set of context-operator associations is defined as the *procedural representation* of an operator sequence. When a procedural representation is available, these learned associations guide immediate operator selection and implicitly promote the application of this specific operator sequence when the task remains unchanged. For example, an operator is followed by another operator in a familiar scenario because the association between the model's internal context, updated by the previous operator, and the next operator is strong. The relationships between operators established through contextual learning can also be flexibly modified due to changes in the upcoming task environment. Thus, while traditional cognitive architectures construct procedures through static, manual assembly of operators, PRIMs adopts a dynamic approach, learning to derive procedures.

Our PRIMs simulation (Ji et al., under revision) of infant's acquisition of lexical and syntactic patterns, as observed in statistical learning (Saffran, Aslin, & Newport, 1996) and algebraic processing (Marcus, Vijayan, Bandi Rao, & Vishton, 1999), shows the same model and learning mechanism can account for both tasks by forming different sets of contextual associations, or procedural representations. In statistical learning tasks, the model learns to identify embedded trisyllabic patterns (e.g., *X-Y-Z* composed of fixed syllables) within continuous syllable sequences using a 3-gram procedure. The 3-gram procedure encodes three syllables to the working memory buffer and detects a working memory-declarative memory mismatch at the fourth. In these tasks, the model can leverage both orderly syllable and operator contexts to predict the subsequent operator. Alternatively, in algebraic tasks, the syllable content within trisyllabic patterns varies. The model identifies repeated patterns (e.g., *a-b-a* composed of syllable classes) using an abstract repetition procedure, which detects an input-working memory match. In this task, the orderly operator context predicts the subsequent operator, but the highly variable syllables do not. This exemplifies how PRIMs can learn unique procedural representations for each task that traditionally would need to be manually defined by multiple separate models.

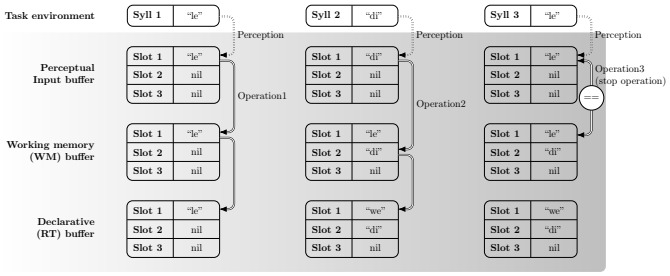
Nevertheless, procedure-based contextual learning, by operating only on the immediate context, prevents operators from forming expectations about the sequence's final outcome. To overcome this limitation, we introduce a skill buffer that incorporates goal-orientation to operator sequences. This buffer enables context-dependent operators to form sequence-based expectations, directing them toward specific outcomes. Potentially, this skill level also enables the coordination of multi-skills necessary for more complex cognitive tasks. This echoes, for example, a previous PRIMs model where visual attention and working memory skills are linked to perform the attentional blink task (Hoekstra, Martens, & Taatgen, 2020).

To intuitively illustrate standard PRIMs' procedural approach and further motivate the need for a skill buffer, this section revisits the "le-di-le" example from Ji et al. (under revision), a concrete instance of the algebraic *a-b-a* pattern (Marcus et al., 1999).

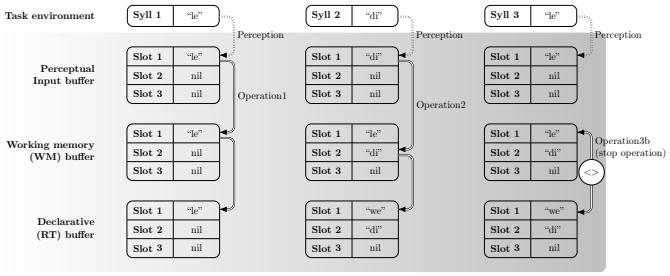
Procedural Representations

In this simulation, it is assumed that the presented syllables is automatically perceived by the input buffer. The input content can then be further encoded into a working memory buffer. In this specific model, working memory encoding is assumed to be sequential. The encoding of working memory starts from the first slot and then sequentially fills the next empty slot. As a simplification, this model does not strictly differentiate between encoding information into working memory and retrieving information from long-term memory. Consequently, when content is encoded into working memory, the model concurrently initiates a declarative memory retrieval process. During this process, a declarative item consistent with the currently encoded position-specific working memory content is selected from long-term memory and placed into the declarative buffer. When multiple possibilities exist, the item with the highest activation level is retrieved. When a suitable declarative item is absent, retrieval failure results, and the retrieval buffer remains empty.

In Figure 2A, three syllables are sequentially presented. Initially, the first syllable "le" is automatically captured by the input buffer. This syllable is then encoded into the first slot of the working memory buffer using encode operator 1. Concurrently, operator 1 initiates a long-term memory retrieval by placing "le" into the first slot of the declarative buffer. Following the application of operator 1, the model's buffer contents are updated. Subsequently, encode operator 2 places the second syllable "di" into the next available slot in the working memory buffer and also triggers a long-term memory re-

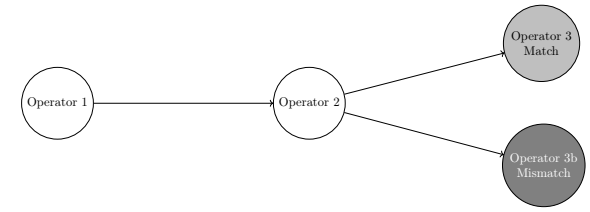


Panel A: The cognitive architecture perceives the syllables "le", "di", and "le" sequentially from the task environment, triggering successive operations 1, 2, and 3. These operations find a match between the input and working memory buffers. This operation sequence is related to the more abstract repetition strategy in learning syntax.

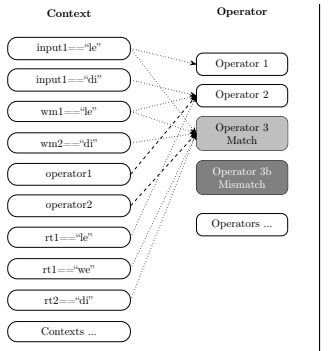


Panel B: The cognitive architecture again perceives the syllables "le", "di", and "le" sequentially. However, this time, it selects alternative operations 1, 2, and 3b, which identify a mismatch between the working memory and declarative buffers. This operation sequence aligns with the more concrete 1-gram strategy in lexical learning. Because the "word-boundary" mismatch occurs at the second syllable, the remaining non-mismatched part constitutes a 1-gram unit.

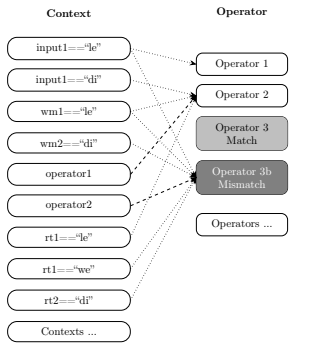
Figure 2: Two procedures for the algebraic *a-b-a* pattern. Reinforcement is applied to context-operator associations regardless of the chosen sequence in the standard approach.



Panel A: Confusion arises in selecting the immediate operator when there are two possible sequences of operators that can perform the same task.



Panel B: Set of context-operator associations corresponding to the syntactic repetition skill.



Panel C: Set of context-operator associations corresponding to the lexical 1-gram skill.

Figure 3: Two different operator sequences and corresponding sets of context-operator associations.

trieval. Note that in this case, the first encoding operator cannot be reselected because the first working memory buffer slot is already occupied. When the third syllable "le" is perceived, the model detects a repetition with the first syllable already stored in the working memory buffer. The detection of repetition triggers a stop operator (describe shortly in details) , terminating the operator sequence and initiating the model's learning mechanism, which updates the associations between model contexts and operators at each step.

In PRIMs, the above operator sequence reaching a particular stop operator is not the only possibility. The model can consider a hypothesis space of possible stop operators. These *stop operators* constitute a subset of generic comparison operators, specifically designed to function as the end of an operator sequence. As described in Ji et al. (under revision), the possible stop operators include comparisons informed by empirically derived sameness or difference strategies (de la Cruz-Pavía & Gervain, 2021). The repetition procedure illustrated above is a particular example of a sameness strategy, which identifies a match between the perceptual input and the previously encoded working memory content (or the declarative content retrieved from long-term memory). Other possibilities include difference strategies. Such a strategy entails procedures that detect a position-specific mismatch between working memory and declarative memory contents (see next section). The procedural representation, however, lacks the capacity to distinguish between context-operator associations that result in different stop operators, opting instead for a uniform update of contextual associations to support immediate context-dependent operator selection.

Skill Representations

Since the operators can initially be applied flexibly, leading to different stop operators, there is at least one other possible approach to processing the simple pattern example. In Figure 2B, the first two steps of the model mirror the previous example. However, the retrieved pattern from long-term memory diverges from the content currently encoded in the working memory buffer. This discrepancy triggers a stop operator that detects a mismatch between the contents of the first slot in the two buffers. This mismatch can signal that the current pattern-level expectation has been violated, indicating both the segmentation of the pattern and the unfolding of the subsequent pattern.

This mismatch operator subsequently terminates the operator sequence and activates the model's learning mechanism.

This operator sequence or procedure should result in a different set of context-operator associations than those learned in the previous example. This creates ambiguity, as depicted in Figure 3A. After operator 2, the model faces a decision: should it select an operator that detects a match or one that detects a mismatch? Distinguishing between these two operator sequences or procedure is crucial, as it allows the model to predict the recognized outcome of the procedure.

However, in the absence of overall expectations regarding operator sequences, the model is unable to anticipate the ultimate consequences of a given operator sequence (e.g., whether a stop operator will result in a match or mismatch) or the recognition the model will make (e.g., syllable repetition or pattern mismatch; see Figure 3A). This limitation can compromise performance, as immediate task demands may inadvertently influence the model's operator selection. For example, the mismatch operator's reliance on the second input, in contrast to the match operator's dependence on a third input, can induce a bias towards shorter 1-gram procedure completion, even when the longer repetition procedure would be a more general interpretation of the task pattern. Thus, this means that learning and discrimination need to occur not only between operators but also between distinct procedures.

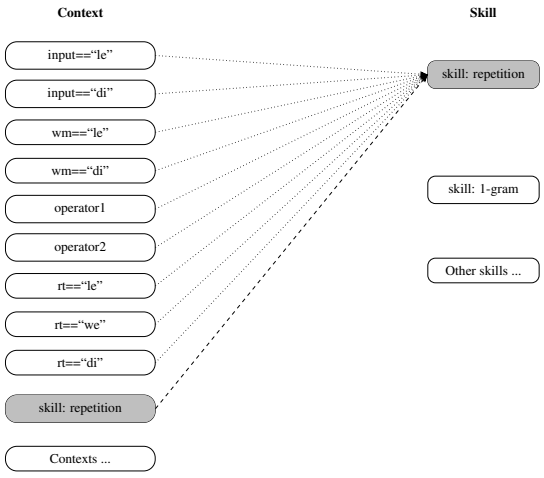


Figure 4: A set of context-skill associations.

Standard PRIMs is equiped with a single procedural representation, allowing one learned set of context-operator associations. However, this example highlights that two procedural representations should serve as pointers to two separate, non-interfering probability models at the operator selection level. This clarifies the necessity of learning and distinguishing between multiple procedural representations (or different sets of context-operator associations). Procedural representations now allow specification of the operator sequence leading to a particular stop operator. These specialized representations, distinct from the undifferentiated procedural representation, are termed *skills*.

In our simulations we enhanced PRIMs with a dedicated skill buffer. This buffer stores a number of chunks, where the chunks are abstracted labels of skill representations. A skill chunk defines two key details: (a) it specifies the particular stop operator that a given operator sequence will lead to; and (b) a skill chunk references to a skill-specific set of context-operator associations like a procedure representation. This allows the model to establish distinct context-operator associations based on the model contexts generated during specific procedures (see Figure 3). Moreover, skills (conveniently abstracted by their labels) can be further differentiated through learned context-skill associations (see Figure 4).

Implication of skills Since skill representations are partly defined by the corresponding stop operators, we can assume that skills are constrained by cognitive architecture. However, to gradually form these context-based skill representations, the model must go through a process of skill discovery and implicit learning. Initially, the task pattern guides random, immediate operator selection, without any specific purpose. When a procedural representation is formed, the cognitive process incorporates immediate contextual information when selecting the next operator. At this point, the result of the selected operator sequence may lead to an operator action that happens to be a stop operator, but there is no expectation of reaching this stop operator. The initial phase is therefore flexible and exploratory. However, when the model repeatedly reaches a certain stop operator and forms a skill representation, a skill expectation for reaching that stop operator is formed and can be considered goal-oriented. Note that the goal-oriented nature of skills does not necessitate explicit awareness. While skill may relate to explicit attention during simulated task processing in a wakeful state, it can also be an implicit process directed towards an imagined goal, such as experiments done among sleeping infants. In this enhanced implementation, the model always selects a skill first and evaluates that skill whenever the model's expectation of the skill's stop operator, based on the actual upcoming stop operator, is either confirmed or violated. Therefore, although the stop operator is the last operator applied, it is the first element in the model's skill expectation. At this point, the individual operators are guided by the skill expectation rather than by random or context-based immediate operator selection.

Application domain The gradual learning of skill representations prepares PRIMs' bottom-up learning approach to find solutions for more complex tasks. Cross-situational word learning is a good example of this. In naturalistic situations, young children need to track consistent mappings of words with specific objects or events across various ambiguous contexts (Yu & Ballard, 2007; Monaghan, Mattock, Davies, & Smith, 2014; Dunn, Frost, & Monaghan, 2024). A specific example is the task designed by Dunn et al. (2024). In their study, distinct pseudowords (e.g., "chelad" or "gorshall") are combined with different morpheme endings (e.g., "-noo" for "it is not the..." or "-tha" for "it is the...") to form a continuous utterance (e.g., "chelad-noo-gorshall-tha", or "gorshall-tha-

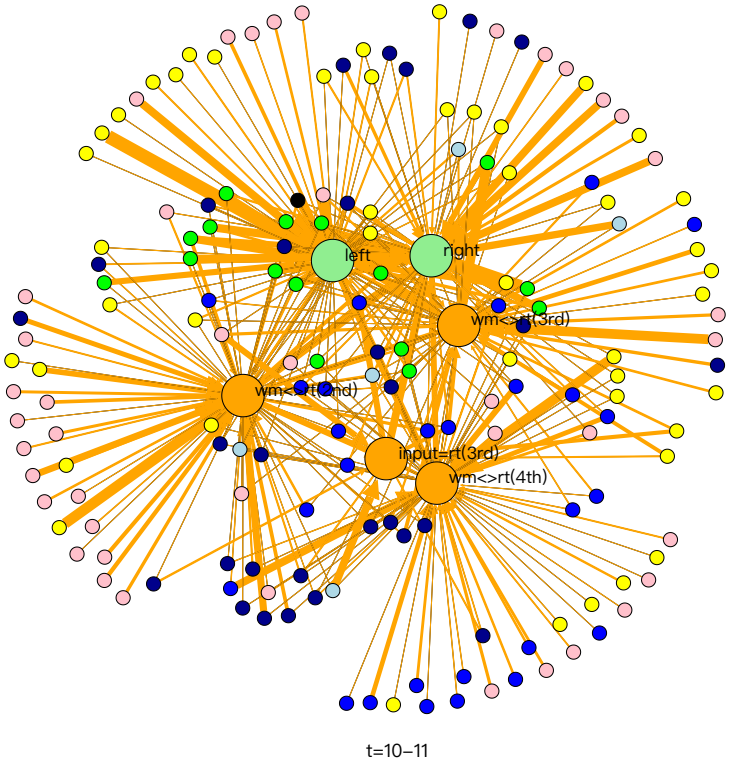


Figure 5: This figure depicts learned context-skill associations following training on a cross-situational word learning task. For convenience, only positive associations are displayed. Circles represent skills: parsing skills (orange), gaze skills (green). Dots represent model contexts: skill (orange), declarative item from the last skill (pink), operator (yellow), input content (light blue), working memory content (blue), retrieval content (dark blue), and background context (black). Edge widths correspond to association weights.

chelad-noo", meaning "it is not the chelad, it is the gorshall"). Concurrently, two images are presented to young children, positioned on either the right or left side, with each image corresponding to one of the pseudowords (e.g., an image for "chelad" and an image for "gorshall"). Cross-situational word learning presents a challenge, as young children cannot be instructed in a top-down manner or rely on explicit rules (Chomsky & Schützenberger, 1959; Pinker, 1999). Instead, they must engage in bottom-up learning, simultaneously parsing utterances into meaningful phonological units (e.g., "chelad-noo" or "gorshall-tha") and linking these units with referents from the environment. Initially, parsing may be partial (e.g., "chelad") or overally redundant (e.g., "chelad-noo-gorshall"), leading to weak associations with random or incorrect referents. However, as learning progresses, stronger and more accurate mappings begin to form between consistent phonological patterns (e.g., "gorshall-tha") and their correct environmental referents (e.g., an image for "gorshall").

Figure 5 illustrates the learned context-skill associations when the PRIMs architecture encounters the particular cross-situational word-learning task (Ji, van Rij, & Taatgen, 2025). This figure reveals skills related to phonological processing aspects of cross-situational word learning, including the detection of syllable repetition and pattern mismatch (orange circles). We can observe that the parsing process, based on pattern mismatch detection, is highly varied, potentially occurring at the second, third, or fourth syllable position (e.g., parse out "chelad", "chelad-noo", or "chelad-noo-gorshall"). Furthermore, the model sometimes exhibits syllable repetition detection (green circles), which can occur if the encoded pattern crosses continuous utterance boundaries (e.g., "gorshall-tha-gorshall"). In addition, skills corresponding to the learning of referents from the environment (i.e., left or right gaze) are evident (large green circles). Crucially, we can see that the model can form more complex skill-skill associations (e.g., parsing associated with right gaze) in the more complex scenario of cross-situational word learning. However, the operator sequence itself is not rule-like, as the left or right gaze is highly dependent on the task context, which in our case forms the model context distributed over the various buffers (colored dots).

Implementing Skill-Based Processing

We now provide implementation descriptions of PRIMs processing. We use pseudocode to illustrate the concept, including the initial selection of skills, as well as the immediate operator selection in subsequent model steps. We then describe the selection of skills and operators in more detail.

Specifically, Pseudocode 1 illustrates the model's processing steps. Initially, the model begins with open trial-and-error

```
while task is ongoing do
    // select skill
    if learned skill label/s[1] available then
        calculate skill activations[3];
        select skill (highest activation);
    end
    // select operator
    while stop operator not reached do
        for a list of all operators do
            calculate operator activations[2,3];
            select operator (highest activation);
            if operator not passing minimal checking
              then
                remove from list of operators
            else
                carry out operator;
                collect model contexts (for operator,
                  skill);
                increment model step
            end
        end
    end
    // label and update
    label current operator sequence as the skill label;
    update context-operator and context-skill
      associations[3];
end
```

Pseudocode 1: PRIMs processing. [1]Section 2 details the implementation of skills, including how they are defined based on skill labels, anticipated outcomes or stop operators, and corresponding context-operator associations. Please see that section for further information. [2]When a skill is available and selected, the model loads the context-operator associations to calculate operator activation. This operator activation is then combined with a random activation noise term. Therefore, operator selection is context-based. However, when no skill is available, the model only applies the random activation noise to calculate operator activation. Operator selection is thus based on trial and error. [3]Note that the learning mechanism, which uniformly adaptes an enhanced discriminative error-driven learning approach (Hoppe et al., 2022), along with the corresponding calculate activation functions, are not elaborated in this paper due to page limits.

operator application, just like the standard approach. When contextual associations are available for inferences, the selection of skills and operators becomes context-based.

Skill selection

Upon the first successful application of a stop operator, an operator sequence terminates, triggering the skill labeling and contextual updating processes (see *label and update* in Pseudocode 1). In subsequent trials, the enhanced approach can then initiate an additional skill selection process at the beginning of each new operator sequence (see *select skill* in Pseudocode 1). When a learned skill representation is available, this initial process selects a representation (e.g., a chunk like "repetition" or "1-gram") and loads it into the skill buffer. The model then applies the corresponding specialized set of context-operator associations consistently throughout the current operator sequence, leveraging these associations to guide operator selection. Skill selection thus inherently possesses a goal-oriented nature, predicting the anticipated stop operator that will eventually conclude the operator sequence. This skill prediction, however, remains a heuristic, which can only be confirmed or contradicted upon the skill's completion. With skill selection, the model can more equitably weigh different operator sequences at a higher level, enabling a trade-off between skill expectations.

The increasing confirmation of a particular skill suggests its suitability for the current task, leading the model to preferentially apply the same essential operator sequence to terminate at the anticipated stop operator. However, if a skill fails to reach its anticipated result, the established skill expectation is violated. Such occurrences are frequent in novel task scenarios, where the minimal conditional checks associated with a mid-way essential operator and/or a stop operator from a previously successful skill no longer hold true. In such instances, while the model continues to predict operators based on learned context-operator associations, a previously highly activated operator can be excluded in a certain changed immediate scenario. This process continues until a suitable less activated operator is selected.

The alternative skill that emerges may be a previously acquired skill or a novel one. If novel, a new skill label and corresponding context-operator association set needs to be created (see *label and update* in Pseudocode 1). A single instance of a successful alternative skill is insufficient to displace an established skill expectation. Consequently, the previous skill is initially retained, and its specialized associations still guide operator selection. However, if the current operator sequence consistently deviates from the anticipated skill, the alternative skill gradually gains prominence. Over time, the consistently applied alternative skill may eventually supersede the previous expectation as the preferred skill for the novel task scenario.

Operator selection

Operator selection is initially flexible, based on activation noise and minimal conditional checking. However, once

context-operator associations are learned, operator selection is based on both these acquired associations and minimal condition checking. An operator's selection (see *select operator* in Pseudocode 1), whether due to initial activation noise or contextually driven by the activation equation, does not guarantee its execution. It is still subject to PRIMs' minimal conditional checking assumption. For instance, consider a primitive operator used to move data from one buffer (input1) to another (wm2). This operator can proceed whenever the source buffer (input1) is not void and the destination buffer (wm2) is vacant.[1] Likewise, a primitive operator comparing the contents of two buffers (input1 and wm2) only requires that both buffers contain some information. If these primitive operators meet the basic criteria of the current model context, they are eligible for selection and execution. Otherwise, the next most highly activated operator is considered. This checking process repeats until a suitable operator is selected or all available operators have been tried and failed. The primary purpose of this minimal condition checking is to filter out impossible events during information processing and to select the most appropriate process based on contextual associations.

Conclusion

Extending PRIMs' open procedural learning, this paper introduces a skill-based approach to distinguish distinct procedures. This enhancement broadens PRIMs' bottom-up perspective to simulate skill-level procedural learning, connecting the open discovery of operator sequences with skill adaptation and multi-skill processing.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press. Retrieved from http://dx.doi.org/10.1093/acprof:oso/9780195324259.001.0001

Chomsky, N., & Schützenberger, M. (1959). The algebraic theory of context-free languages. In *Studies in logic and the foundations of mathematics* (Vol. 26, p. 118–161). Elsevier. Retrieved from http://dx.doi.org/10.1016/s0049-237x(09)70104-1

de la Cruz-Pavía, I., & Gervain, J. (2021, February). Infants' perception of repetition-based regularities in speech: a look from the perspective of the same/different distinction. *Current Opinion in Behavioral Sciences*, *37*, 125–132. Retrieved from http://dx.doi.org/10.1016/j.cobeha.2020.11.014

Dunn, K. J., Frost, R. L., & Monaghan, P. (2024). Infants' attention during cross-situational word learning: Environmental variability promotes novelty preference. *Journal of Experimental Child Psychology*, *241*, 105859.

Hoekstra, C., Martens, S., & Taatgen, N. A. (2020, jul). A skill-based approach to modeling the attentional blink. *Topics in Cognitive Science*, *12*(3), 1030–1045. Retrieved from http://dx.doi.org/10.1111/tops.12514

Hoppe, D. B., Hendriks, P., Ramscar, M., & van Rij, J. (2022, January). An exploration of error-driven learning in simple two-layer networks from a discriminative learning perspective. *Behavior Research Methods*, *54*(5), 2221–2251. Retrieved from http://dx.doi.org/10.3758/s13428-021-01711-5

Ji, Y., van Rij, J., & Taatgen, N. (under revision). *Simulating procedure discovery in early language acquisition: Domain-general cognition with contextual learning.*

Ji, Y., van Rij, J., & Taatgen, N. (2025). How do children acquire syntactic structures? a computational simulation of learning syntax from input. In *International workshop on the acquisition of (syntactic) complexity at the interface (march 26-27, 2025).*

Laird, J. E., Lebiere, C., & Rosenbloom, P. S. (2017, December). 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. Retrieved from https://doi.org/10.1609/aimag.v38i4.2744

Marcus, G. F., Vijayan, S., Bandi Rao, S., & Vishton, P. M. (1999, January). Rule learning by seven-month-old infants. *Science*, *283*(5398), 77–80. Retrieved from http://dx.doi.org/10.1126/science.283.5398.77

Monaghan, P., Mattock, K., Davies, R. A. I., & Smith, A. C. (2014, October). Gavagai is as gavagai does: Learning nouns and verbs from cross-situational statistics. *Cognitive Science*, *39*(5), 1099–1112. Retrieved from http://dx.doi.org/10.1111/cogs.12186

Newell, A., Shaw, J. C., & Simon, H. A. (1958). Elements of a theory of human problem solving. *Psychological Review*, *65*(3), 151–166. Retrieved from http://dx.doi.org/10.1037/h0048495

Pinker, S. (1999, January). Out of the minds of babes. *Science*, *283*(5398), 40–41. Retrieved from http://dx.doi.org/10.1126/science.283.5398.40

Saffran, J. R., Aslin, R. N., & Newport, E. L. (1996, December). Statistical learning by 8-month-old infants. *Science*, *274*(5294), 1926–1928. Retrieved from http://dx.doi.org/10.1126/science.274.5294.1926

Taatgen, N. A. (2013, July). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471. Retrieved from http://dx.doi.org/10.1037/a0033138

Yu, C., & Ballard, D. H. (2007, August). A unified model of early word learning: Integrating statistical and social cues. *Neurocomputing*, *70*(13–15), 2149–2165. Retrieved from http://dx.doi.org/10.1016/j.neucom.2006.01.034

[1] Note that the current model assumes these encoding operators orderly encode an incoming stream of inputs into the working memory buffer to form a sequence. Nevertheless, these operators remain general-purpose, as they do not require the modeler to predefine specific conditions, such as the exact content of the sequence.

Learning Dynamics in Anxiety: The Role of Punishment Sensitivity and Learning Rate in Sequential Evaluation

Alicia Muñiz-Jiménez (ali.psi.neuro@gmail.com)
Department of Psychology, National Autonomous University of Mexico (UNAM), Mexico City, Mexico.

Víctor Mijangos (vmijangosc@ciencias.unam.mx)
Department of Science, National Autonomous University of Mexico (UNAM), Mexico City, Mexico.

Arturo Bouzas (arbouria@unam.mx)
Department of Psychology, National Autonomous University of Mexico (UNAM), Mexico City, Mexico.

Abstract

Anxiety is fundamentally an anticipatory response to uncertain future threats, making sequential evaluation crucial. However, most computational research has involved one-step tasks that identify an elevated punishment learning rate as a key feature of anxious behavior, while some studies also suggest a lack of evidence for the role of the punishment sensitivity parameter. This contradicts recent research involving sequential evaluation tasks, which supports the idea that anxiety is primarily an uncertainty disorder. Nevertheless, these experiments have mainly focused on model-based algorithms and have left the role of the punishment learning rate unexplored. To reconcile the differences in the literature and better understand how these two parameters influence anxiety, two hybrid models were developed: one with differentiated learning rates for rewards and punishments, and another with differentiated sensitivity parameters. The models were then evaluated in the Cliff Walking task, using a deterministic environment and incorporating stochasticity in action selection to simulate that an agent lacks complete control over its decisions. The results suggest that the impact of estimated punishments on planning is more significant than the speed at which they are learned, highlighting the importance of heightened punishment sensitivity in anxiety-related behaviors such as avoidance, risk aversion, threat overestimation, and fear generalization.

Keywords: anxiety; reinforcement learning; computational psychiatry; sequential evaluation

Introduction

Computational modeling for studying anxiety is of great importance as anxiety disorders have the highest incidence worldwide among mental disorders (Stein, Scott, de Jonge, & Kessler, 2017). Specifically, the reinforcement learning (RL) approach has aimed to construct mechanistic models that explain the underlying components and their interactions involved in anxious decision-making (Raymond, Steele, & Seriès, 2017; Pike & Robinson, 2022), where an altered part of the model describes anxious behavior (Huys, 2014), for example, a higher value in a certain parameter. However, there are mixed results regarding whether a higher punishment learning rate or an enhanced punishment sensitivity parameter plays a key role in anxious decision-making. This discrepancy becomes even more relevant when considering that these different conclusions stem from distinct kinds of tasks and, consequently, different types of models.

On one hand, most anxiety computational research has focused on simplified one-step tasks, such as multi-armed bandits, where an agent performs many trials but each trial ends after a single action, meaning the agent only needs to learn the appropriate response for a single environmental state (Yamamori & Robinson, 2023). Additionally, a systematic review and meta-analysis of 27 articles by Pike and Robinson (2022) supports that, in studies using one-step tasks, an enhanced punishment learning rate is the primary factor for modeling anxious decision-making, while also indicating a lack of evidence for the role of punishment sensitivity. Still, due to the nature of these tasks, planning has not been considered and the models developed only try to replicate direct learning in one-state environments. Nevertheless, anxiety is an anticipatory response to uncertain future threats perceived as uncontrollable and highly aversive (Clark & Beck, 2010), involving a multi-step process in which it should be considered how planning may be biased (Sharp, 2025).

On the other hand, anxiety computational research involving sequential evaluation tasks is more recent and as a result comprises only a few studies, which primarily rely on variations of the value iteration algorithm to model biased offline planning and reproduce common behaviors observed in many anxiety disorders, such as avoidance, risk aversion, threat overestimation, and fear generalization (Zorowitz, Momennejad, & Daw, 2020; Gagne & Dayan, 2022). These works have found that heightened punishment sensitivity is a key feature to reproduce anxious behavior, highlighting that anxiety is mainly a disorder of uncertainty (Brown, Price, & Dombrovski, 2023), although the role of the punishment learning rate remains unexplored in multi-step tasks.

Therefore, while earlier studies on modeling anxious behavior mainly emphasize the importance of how quickly punishment information is integrated over time, what matters in sequential evaluation experiments is how much an agent anticipates and dislikes being punished. As computational research on anxiety is relatively recent, no study has tested the contributions of both parameters —punishment learning rate and punishment sensitivity— simultaneously in a sequential evaluation task, which is a more natural setting for anxiety. To address this, it is necessary to use hybrid models that retain the core components of both approaches: direct learning in the environment and offline planning. Ultimately, this will help reconcile differences in the literature and offer deeper insights into the learning dynamics of anxiety in sequential evaluation tasks.

Among hybrid RL models, empirical evidence suggests that a hybrid successor representation (SR) is the most effec-

tive algorithm for replicating human decision-making in sequential evaluation tasks (Momennejad et al., 2017; Russek, Momennejad, Botvinick, Gershman, & Daw, 2017). Specifically, SR-Dyna is considered the best model for capturing human retrospective revaluation behavior (Momennejad, 2020). Moreover, online planning is employed in the basic SR algorithm when successor representations are estimated or used to compute the estimation of another variable (Gershman, 2018). Therefore, the SR-Dyna model implements direct learning in the environment, online planning, and offline planning. The integration of these three mechanisms —with some modifications— may better reflect how individuals with anxiety process information in real-world situations, compared to models that incorporate only one or two of them. For example, for a woman with acrophobia, direct learning instantly increases her association of heights with threat after falling from a cliff. Meanwhile, offline planning involves rumination on various possible situations based on past experiences while at rest. Finally, online planning manifests as hypervigilance for potential future threats when she's actually at heights and has to consider path estimations, with all three mechanisms reinforcing her fear of heights.

Furthermore, in SR family models, rewards are learned independently from successor representations (trajectories), but both can later be used to compute Q-values for each state in an environment (Ducarouge & Sigaud, 2017). In particular, this independence in learning allows for specific modifications to the model's equations to assess the impact of differentiated learning rates and sensitivity parameters, taking into account distinct learning dynamics. For example, a first model can be constructed by implementing differentiated learning rates in the reward function. In contrast, a second model can be designed with differentiated sensitivity parameters for rewards and punishments in the successor function. Next, simulations using different values of these parameters can assess the impact on anxious behavior of a higher punishment learning rate in the first model and an enhanced punishment sensitivity parameter in the second one. Hence, we propose using an SR-Dyna algorithm as a basis to design new models to address the objective of this work.

Methodology

The present work adopts a theoretical modeling approach to generate predictive insights for anxiety research. Two models were developed to simulate agents with varying levels of punishment learning rate or punishment sensitivity, with the aim of evaluating their respective contributions to anxious decision-making within a Markov Decision Process (MDP).

Models

In algorithm 1, we present the pseudocode for an SR-Dyna model considering a deterministic environment, incorporating temporal difference (TD) learning and estimating Q-values, followed by a detailed explanation of the modifications introduced to it for the first and second models proposed in this work.

Algorithm 1 : SR-Dyna

1: **Initialize** $R(s)\ \forall\ s \in S$.
2: **Initialize** $M(s,s',a)\ \forall\ s,s' \in S$ and $a \in A(s)$.
3: **Initialize** $Q(s,a)\ \forall\ s \in S$ and $a \in A(s)$.
4: **Initialize** $E(s,a)\ \forall\ s \in S$ and $a \in A(s)$.
5: **while** episode $\leq N$ **do**
6: $s \leftarrow$ actual state
7: $a \leftarrow \varepsilon$-greddy(s,Q)
8: Execute action a; observe the received reward (r), and the next state (s').
9: $a^+ \leftarrow \arg\max_a Q(s',a')$
10: $M(s,\mathbf{s}',a) \leftarrow M(s,\mathbf{s}',a) + \alpha(I_{[s=s']} + \gamma M(s',\mathbf{s}'',a^+) - M(s,\mathbf{s}',a))$
11: $R(s) \leftarrow R(s) + \alpha(r - R(s))$
12: $Q(s,a) \leftarrow \sum_{s'} M(s,\mathbf{s}',a)R(s')$
13: $E(s,a) \leftarrow r,s'$
14: **for** n **do**
15: $s \leftarrow$ previously observed random state.
16: $a \leftarrow$ random action previously executed in s.
17: $r,s' \leftarrow E(s,a)$
18: Execute steps from **9** to **12** using the new values for s, a, r and s'.
19: **end for**
20: **end while**

Parameters: N, ε, α, γ and n.

First, in Algorithm 1 the vector for rewards (R), the successor tensor (M), the Q-value matrix (Q) and the matrix with the environment's model (E) are initialized with zeros in each entry. Then, during each episode in the model-free part of the model, the agent selects an action (a) using an ε-greedy policy, executes it, receives a reward (r) and observes the next state (s'). Later, the agent updates the current successor representation using the TD method, incorporating both immediate and discounted future occupancy predictions. Additionally, the estimated reward, Q-value estimates, and the environment's model are updated. In the Dyna-style offline planning phase (steps 14-18), the algorithm replays stored experiences using the environment's model and updates the SR model estimations.

Model 1: Dyna α-SR This model differs from Algorithm 1 solely in how the learning rate (α) is selected in the reward function (step 11). Specifically, the α value now depends on whether the agent receives a reward or a punishment, as shown in Equation 1, in order to have differentiated learning rates.

$$\alpha = \begin{cases} \alpha^+ & \text{if } r \geq 0 \\ \alpha^- & \text{if } r < 0 \end{cases} \tag{1}$$

Model 2: Dyna β-SR Building on Algorithm 1 and the work of Zorowitz et al. (2020) to model anxious behavior, we introduce a novel variant: Dyna β-SR. In this new model, it is necessary not only to calculate the action with the highest value (a^+) for the next state, but also the lowest (a^-), as

shown in Equations 2 and 3.

$$a^{+} \leftarrow \arg\max_{a} Q(s', a') \tag{2}$$

$$a^{-} \leftarrow \arg\min_{a} Q(s', a') \tag{3}$$

Later, a^{+} and a^{-} are used in the β^{M} module (Eq. 4) to estimate the successor representations with the highest and lowest action for the next state, which are multiplied by ω and its complement, respectively. Therefore, we can say that a pessimistic agent ($\omega = 0$) is sensitive to the paths that lead to punishments or threats, considering that it lacks complete control over its actions. This can be considered as a low level of self-efficacy, reflecting a lack of belief in their ability to perform the necessary behaviors to achieve specific performance goals (Bandura, 1978; Zorowitz et al., 2020). In contrast, an optimistic agent ($\omega = 1$) anticipates that its future actions will fully align with its preferences to maximize rewards (Zorowitz et al., 2020), being sensitive to the paths that lead to them. Finally, β^{M} is used in the successor function (Eq. 5) as the successor representation for the next state.

$$\beta^{M} \leftarrow \omega M(s', \mathbf{s}'', a^{+}) + (1-\omega) M(s', \mathbf{s}'', a^{-}) \tag{4}$$

$$M(s, \mathbf{s}', a) \leftarrow M(s, \mathbf{s}', a) + \alpha \left(I_{[s=s']} + \gamma \beta^{M} - M(s, \mathbf{s}', a) \right) \tag{5}$$

Experimental Task

A variation of the Cliff Walking task (Fig. 1) was implemented in a 9x9 gridworld with a discrete-time environment, finite horizon, and deterministic dynamics. The stochasticity lies in the agent's action selection, as it follows an ε-greedy policy to choose the action it executes in each state. This policy is used both during the agent's training phase ($N = 200$ episodes) and in the final test (1 rollout) to simulate that the agent does not have complete control over its actions, as is the case in many real-world scenarios.

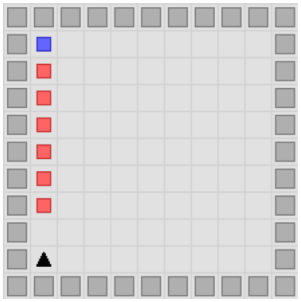


Figure 1: **Virtual environment of the Cliff-Walking task.** The triangle indicates the agent's starting position, the blue square marks the goal, the red squares represent aversive states (the cliff), and the light gray squares are neutral, where neither a reward nor a punishment is received.

The agent's objective in this environment is to reach the goal without falling into an aversive state. At the beginning, the agent is unfamiliar with the environment and must explore it to find the optimal route. The agent can perform four possible actions: moving up, down, right, or left, as long as it does not collide with a wall (dark gray squares). Finally, an episode ends under the following conditions: the agent reaches the goal (reward = 1), falls into an aversive state (punishment = -1), or the maximum step limit per episode is reached (100).

Additionally, the anxious behaviors we aimed to evaluate in the experimental task were operationalized as follows:

- **Threat overestimation:** states that would otherwise be neutral are estimated as signaling threat unrealistically.
- **Fear generalization:** neutral states distant from aversive ones are considered dangerous.
- **Avoidance:** the agent consistently moves away from aversive states.
- **Risk aversion:** instead of exploring more of the environment, the agent exploits a safe zone of the state space.

Simulations

Three simulations were conducted for the Dyna α-SR agents with α^{-} values of 0.05, 0.075, and 0.1, respectively, while keeping $\alpha^{+} = 0.05$ constant in all cases. In addition, we ran three simulations for the Dyna β-SR agents with ω values of 1.0, 0.5, and 0.0, respectively. The parameters shared between the models were set as follows: $\alpha = 0.05$ (for the successor function), $\gamma = 0.9$, $\varepsilon = 0.2$ and $n = 3$ (recall). All simulations were performed using the NeuroNav library (Juliani, Barnett, Davis, Sereno, & Momennejad, 2022) as a foundation for programming the agents and the experimental task. The complete code is publicly available at the following link: `https://github.com/Alicia-MJ/RL-Anxiety-MathPsy-ICCC.git`

Results

For each agent, a heatmap of the environment was generated based on the final estimated value of each state, computed by averaging its corresponding Q-values. In these maps, values close to 1.0 (the maximum possible value) appear in deep blue, values near 0 are shown in white, and values close to -1.0 (the minimum possible value) are displayed in dark red. Arrows indicate the highest-value action estimated for each state, while the rollout corresponding to the evaluation phase is highlighted with black arrows. Additionally, as a support to the main analysis, the total number of steps and the total return received per episode were graphed for each agent to evaluate its evolution across episodes. Together, these three analyses provide a more comprehensive assessment of each agent's behavior and learning.

Dyna α-SR Agents

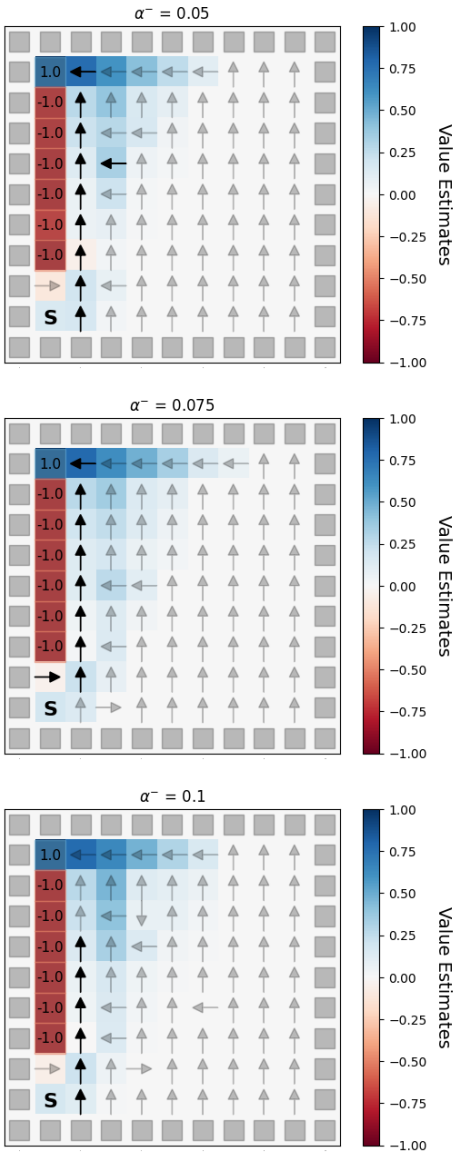


Figure 2: **Dyna α-SR agents' environment estimations.** The top image corresponds to the agent with $\alpha^- = 0.05$ (reward and punishment learning rates are equal), the middle to the agent with $\alpha^- = 0.075$ (50% higher than α^+), and the bottom to the agent with $\alpha^- = 0.1$ (double than α^+).

The three Dyna α-SR agents show a very similar behavior across the three analyses conducted. First, Figure 2 demonstrates that their environmental estimations are highly similar, even in the case where an agent had a punishment learning rate (α^-) that was twice the value of the reward learning rate (α^+). Additionally, all three agents chose a short but unsafe path to the goal during the testing phase (indicated by black arrows), underestimating the danger. While the first and second agents managed to reach the goal in the rollout, the third agent followed a similar path but, due to the stochasticity in action selection, ended up falling into the cliff.

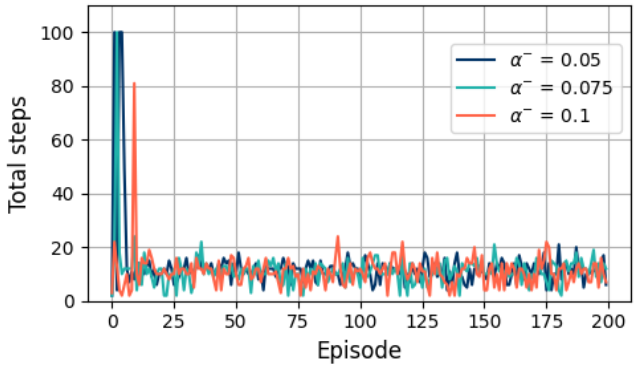


Figure 3: **Dyna α-SR agents' total steps per episode.**

Moreover, Figure 3 shows that, in general, the agents used a similarly small number of temporal steps across episodes, which is consistent with their preference for the shortest path to the goal. Only in a few episodes did the agents use more temporal steps than usual, probably because they ended up in an area that was still unexplored and farther from the cliff. Additionally, due to the similarities in both environmental estimations and the number of steps per episode, a similar level of goal achievement was expected. This is confirmed in Figure 4, which indicates that the agents mainly oscillated between reaching the goal (total return = 1) and falling into the cliff (total return = –1) across episodes, reflecting the inherent risk of choosing the shortest path. Finally, minor differences in behavior between the agents can be attributed to the stochasticity of the programmed policy.

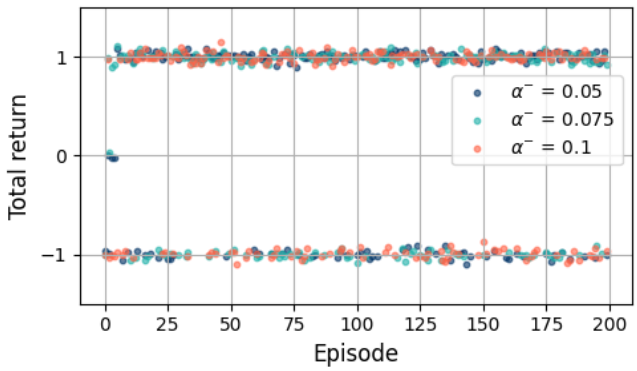


Figure 4: **Dyna α-SR agents' total return per episode.**

In summary, the results for the Dyna α-SR agents show that a higher punishment learning rate did not lead to the emergence of anxiety-related behaviors such as threat overestimation, fear generalization, avoidance, or risk aversion in the experimental task.

Dyna β-SR Agents

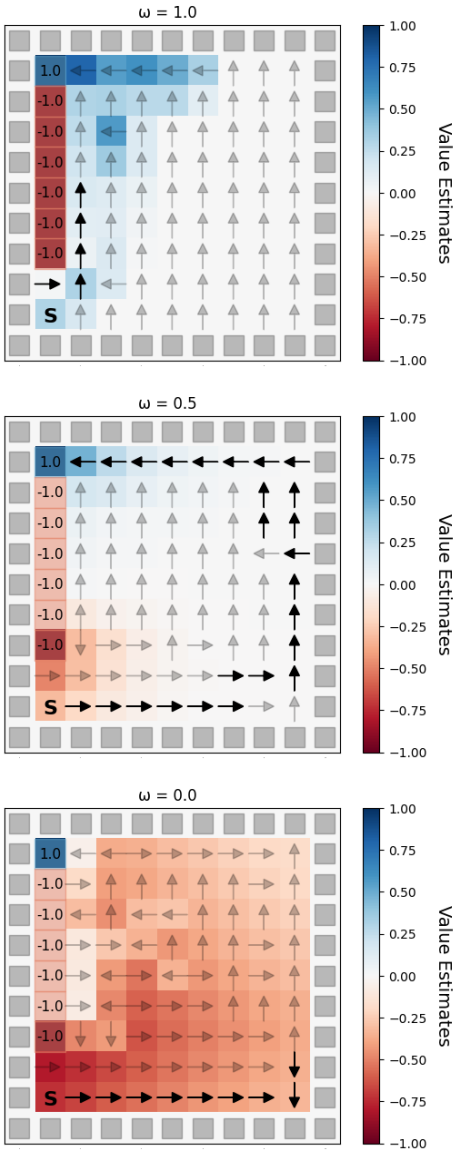


Figure 5: **Dyna β-SR agents' environment estimations**. The top image corresponds to the optimistic agent (ω = 1), the middle to the agent with ω = 0.5, and the bottom to the pessimistic agent (ω = 0).

In contrast to Dyna α-SR agents, Dyna β-SR agents show a marked difference in learning and behavior across the analyses conducted. To begin with, Figure 5 shows that the optimistic agent (ω = 1) learned to take a short but risky route to the goal near the cliff by estimating nearby states positively, without accounting for the possibility of not always being able to choose its own actions. Conversely, the moderately optimistic–pessimistic agent (ω = 0.5) avoided the cliff during the testing phase, as it evaluated some nearby states as dangerous. Nevertheless, it was still able to reach the goal by taking a longer but safer path. Finally, the pessimistic agent (ω = 0) estimated much of the environment as aversive and tried to stay away from the cliff during the rollout, but ultimately failed to reach the goal.

Therefore, although the second and third agents exhibited avoidance behavior, it was the pessimistic one that excessively overestimated threat and demonstrated fear generalization. Additionally, the results suggest that threat overestimation and fear generalization bias reward evaluation, leading an agent to prioritize danger avoidance over goal pursuit. This reflects risk aversion, as shown by the third agent that exploited a safe zone of the state space instead of exploring the environment further.

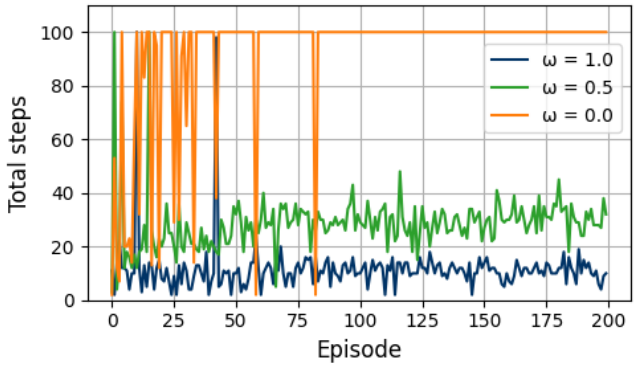


Figure 6: **Dyna β-SR agents' total steps per episode.**

Furthermore, the results from Figure 5 align with those in Figure 6, which show that the optimistic agent (blue) used the fewest temporal steps in most episodes by taking a more direct route to the goal. It was followed by the moderately optimistic–pessimistic agent (green), which took a longer but safer path to the goal. Meanwhile, the pessimistic agent (orange) used the maximum number of steps (100) in most episodes, as it tended to remain in neutral states far from danger.

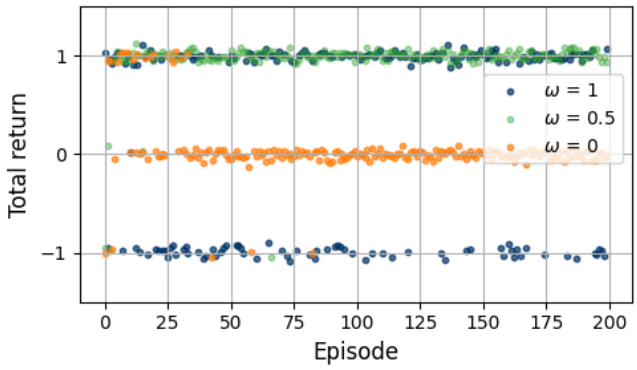


Figure 7: **Dyna β-SR agents' total return per episode.**

Lastly, the patterns observed for each agent in Figure 7 complement the previous results by highlighting how the

agents' environment estimations and learned strategies impacted their actual outcomes. The optimistic agent (blue) alternated between reaching the goal and falling into the cliff. The agent with $\omega = 0.5$ (green) achieved the goal most consistently, ending in neutral or aversive states only occasionally. In contrast, the pessimistic agent (orange) remained in neutral states for most of the episodes, particularly in the last 100.

In summary, the results for the Dyna β-SR agents indicate enhanced punishment sensitivity was directly linked to the exhibition of anxious behavior in the experimental task, as well as the necessity of a level of punishment sensitivity for an agent to identify a safer path to the goal.

Discussion

In this work, we expanded anxiety computational research in sequential evaluation tasks by using hybrid reinforcement learning models to assess the influence of punishment learning rate and punishment sensitivity in order to better understand the learning dynamics underlying anxious behavior. The results suggest that the impact of estimated punishments on planning is more significant than the speed at which they are learned, highlighting the importance of heightened punishment sensitivity in anxious decision-making.

Specifically, the behavioral similarity observed among Dyna α-SR agents indicates that a higher punishment learning rate does not promote the emergence of anxious behavior in the experimental task. In contrast, the performance differences among the Dyna β-SR agents support the idea that increased sensitivity to punishment facilitates the emergence of anxiety-related behaviors, such as avoidance, risk aversion, threat overestimation, and fear generalization. The Dyna β-SR model, in particular, predicts that pessimistic planning —arising from an enhanced sensitivity to threat due to the agent's underestimation of its own self-efficacy or coping resources— may play a central role in the development of these behaviors. This interpretation stems from the observation that what matters most is not how quickly punishments are learned, but how their estimations influence the updating of successor representations, which can lead to a progressive distortion of the cognitive map built by the agent when it has high sensitivity to punishments. In essence, the model highlights the role of catastrophic thinking or worry in anxiety, conceptualized here as a tendency to prioritize the most disadvantageous routes during both online and offline planning.

These findings align with the perspective presented by Brown et al. (2023), as well as with the sequential evaluation studies by Gagne and Dayan (2022); Zorowitz et al. (2020), which suggest that anxiety is primarily a disorder of uncertainty learning, where heightened punishment sensitivity might be a key feature in predicting anxious behavior. On the other hand, the results challenge aspects of prior research based on one-step tasks (Pike & Robinson, 2022), which have emphasized the role of rapid punishment integration in the emergence of anxious decision-making. This raises the question of whether such tasks are adequate for studying anxiety, as they might not fully capture its complexity and could bias our understanding. Alternatively, it could be considered that different task structures —one-step versus multi-step— help clarify the specific ways in which a high punishment learning rate contributes to anxiety. For example, a high learning rate for punishments may be more relevant in the short term, as observed in one-step tasks, while it might lose relevance in sequential evaluation, where a high punishment sensitivity parameter becomes much more significant. Nevertheless, given that anxiety is primarily an anticipatory response to uncertain future threats perceived as uncontrollable and highly aversive (Clark & Beck, 2010), findings from sequential evaluation tasks may provide a more accurate understanding of the factors influencing anxious behavior.

Finally, this work provides valuable insights into the learning dynamics of anxiety considering sequential evaluation, although future research should explore other experimental tasks where differentiated learning rates may play a more significant role, particularly in non-deterministic environments. To achieve this, it will be essential to extend the models' offline planning module to account for uncertainty in environmental dynamics. Additionally, it will be interesting to determine the number of replays an agent makes, which can be linked to the degree of pessimism or surprise. Ultimately, an important next step should be to evaluate the Dyna β-SR model's predictions against human data to assess its plausibility beyond simulation.

Acknowledgements

We thank Dr. Nathaniel Daw and his lab members for their valuable feedback on the research, as well as Dr. Carlos Velázquez, PhD candidate José Luis Baroja, and the three reviewers for their constructive feedback on the paper. Additionally, this work was supported by grants from the CONAHCYT project A1-S-11703, the PITAAE 2024 fellowship, and the DGAPA-PAPIIT projects BG101721 and TA100924.

References

Bandura, A. (1978). Self-efficacy: Toward a unifying theory of behavioral change. *Advances in Behaviour Research and Therapy*, *1*(4), 139-161.

Brown, V. M., Price, R., & Dombrovski, A. Y. (2023). Anxiety as a disorder of uncertainty: implications for understanding maladaptive anxiety, anxious avoidance, and exposure therapy. *Cognitive, Affective, & Behavioral Neuroscience*, *23*(3), 844–868.

Clark, D. A., & Beck, A. T. (2010). *Cognitive therapy of anxiety disorders: Science and practice*. New York, NY: The Guilford Press.

Ducarouge, A., & Sigaud, O. (2017). The successor representation as a model of behavioural flexibility. In *Journées francophones sur la planification, la décision et l'apprentissage pour la conduite de systèmes.* Caen, FR: HAL.

Gagne, C., & Dayan, P. (2022). Peril, prudence and planning as risk, avoidance and worry. *Journal of Mathematical Psychology*, *106*, 102617.

Gershman, S. J. (2018). The successor representation: Its computational logic and neural substrates. *Journal of Neuroscience*, *38*(33), 7193–7200.

Huys, Q. (2014). Computational psychiatry. In D. Jaeger & R. Jung (Eds.), *Encyclopedia of computational neuroscience.* New York, NY: Springer New York.

Juliani, A., Barnett, S., Davis, B., Sereno, M., & Momennejad, I. (2022). Neuro-nav: A library for neurally-plausible reinforcement learning. In *The 5th multidisciplinary conference on reinforcement learning and decision making.* Providence, RI.

Momennejad, I. (2020). Learning structures: Predictive representations, replay, and generalization. *Current Opinion in Behavioral Sciences*, *32*, 155–166.

Momennejad, I., Russek, E. M., Cheong, J. K., Botvinick, M. M., Daw, N. D., & Gershman, S. J. (2017). The successor representation in human reinforcement learning. *Nature Human Behaviour*, *1*(9), 680–692.

Pike, A. C., & Robinson, O. J. (2022). Reinforcement Learning in Patients With Mood and Anxiety Disorders vs Control Individuals: A Systematic Review and Meta-analysis. *JAMA Psychiatry*, *79*(4), 313–322.

Raymond, J. G., Steele, J. D., & Seriès, P. (2017). Modeling trait anxiety: From computational processes to personality. *Frontiers in Psychiatry*, *8*, 1.

Russek, E. M., Momennejad, I., Botvinick, M. M., Gershman, S. J., & Daw, N. D. (2017). Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLoS Computational Biology*, *13*(9), e1005768. doi: 10.1371/journal.pcbi.1005768

Sharp, P. B. (2025). Anxiety involves altered planning. *Trends in Cognitive Sciences*, *29*(2), 118-121.

Stein, D. J., Scott, K. M., de Jonge, P., & Kessler, R. C. (2017). Epidemiology of anxiety disorders: from surveys to nosology and back. *Dialogues in Clinical Neuroscience*, *19*(2), 127–136.

Yamamori, Y., & Robinson, O. J. (2023). Computational perspectives on human fear and anxiety. *Neuroscience and Biobehavioral Reviews*, *144*, 104959.

Zorowitz, S., Momennejad, I., & Daw, N. D. (2020). Anxiety, avoidance, and sequential evaluation. *Computational Psychiatry (Cambridge, Mass.)*, *4*, 10.1162/cpsy_a_00026.

Rule Extraction from Large Language Models within the Clarion Cognitive Architecture

Joseph Killian Jr. (killij4@rpi.edu)
Rensselaer Polytechnic Institute Troy NY 12180, USA

Ron Sun (dr.ron.sun@gmail.com)
Rensselaer Polytechnic Institute
Troy NY 12180, USA

Abstract

Computational cognitive architectures are useful tools for capturing the structures and processes of the mind computationally as well as for simulating behavior. One such cognitive architecture, Clarion, incorporates a two-level structure consisting of both symbolic and sub-symbolic representations (at the two different levels respectively) and bottom-up learning that goes from sub-symbolic to symbolic representations (using the Rule-Extraction-Refinement algorithm). This work explores the integration of Large Language Models (LLMs) into the Clarion framework to enhance its capabilities. The present paper specifically explores a new rule extraction method within Clarion, with SBERT (Sentence-BERT) incorporated into the bottom level of Clarion and a sliding window approach to extract n-gram rules for the top level of Clarion. This modified version of the Rule-Extraction-Refinement algorithm is used to carry out bottom-up learning within the new Clarion. Ongoing experiments on the Pennebaker and King essays dataset (for personality prediction) demonstrate the potential for improved performance and increased explainability when incorporating LLMs into Clarion.

Keywords: Clarion, LLMs, Cognitive Architectures

Introduction

Strong claims on Large Language Models (LLMs) capable of human-like cognition are being made, for example, by DeepSeek and OpenAI (DeepSeek-AI et al., 2025). However, LLMs continue to have problems with trustworthiness, explainability, and so on, even with their progress on many different benchmarks (Romero, Zimmerman, Steinfeld, & Tomasic, 2023). These models include mechanisms like attention that, despite the name, does not reflect actual human attention (Zhao, Xu, & Gao, 2024). Not grounded in human cognition, it may not be possible to create computational models that accurately reflect human cognition (Sun, 2024a).

On the other hand, cognitive architectures have been designed to address such issues. They are computational models that aim to capture the human mind's structure, mechanisms, representations, and processes (Sun, 2024b). They are psychologically realistic models based on existing empirical and theoretical findings on the human mind (human psychology). One such model is the Clarion computational cognitive architecture. This cognitive architecture is unique in some ways. First, each of its main subsystems is centered on a two-level structure, with the top level using symbolic representation and the bottom level using sub-symbolic representation (Sun, 2016). Second, Clarion accentuates bottom-up learning, which goes from sub-symbolic to symbolic representation (through rule extraction). This allows the top-level symbolic representation to emerge through information from the sub-symbolic representation at the bottom level (as will be detailed later).

These features are important in that they provide computational benefits in terms of performance, speed of learning, transfer to other tasks, and synergy between symbolic and sub-symbolic representation, as amply demonstrated by previous work in psychological experimental tasks such as process control, sequential reaction, and minefield navigation (Sun, Merrill, & Peterson, 2001; Sun & Peterson, 1998). These features of Clarion have also been psychologically justified in previous work (Sun et al., 2001; Sun, Slusarz, & Terry, 2005).

However, the question remains whether these benefits will apply if the bottom level of Clarion is composed of an LLM instead of simpler, traditional neural networks. This leads to the contributions of the present paper:

- Incorporating SBERT into Clarion at its bottom level and performing bottom-up learning.
- Developing a rule extraction algorithm with a sliding window n-gram approach for bottom-up learning.
- Presenting some preliminary results that suggest potential for better performance and explainability from rule extraction.

Background

This section provides some background on cognitive architectures and some other relevant work.

Cognitive Architectures

As mentioned earlier, cognitive architectures are psychologically realistic computational models. They aim to be maximally expressive with as few free parameters as possible while adhering to the psychological constraints found in humans (Sun, 2008). Some of the better-known cognitive architectures include ACT-R (Anderson, Matessa, & Lebiere, 1997), Soar (Laird, 2019), and Clarion (Sun, 2016). Each of these models functions as slightly different theories of human cognition (psychology).

This focus on psychological realism is a useful constraint as it forces the designer of such a system to be careful with the details and design decisions (Hélie & Sun, 2014) instead of using whichever components may perform well (as is the case with LLMs) so that the resulting system can better capture human cognition and psychology.

Clarion consists of four subsystems each composed of a two-level structure: a symbolic top level and a sub-symbolic bottom level. The Action-Centered Subsystem (ACS) takes inputs from the environment and generates action decisions (based on its procedural knowledge). The Non-Action Centered Subsystem (NACS) stores factual (declarative) knowledge and is for reasoning and inference. The Motivational Subsystem (MS) deals with basic human motives and their interactions. The Metacognitive Subsystem (MCS) uses different modules to monitor and regulate cognitive processes. These subsystems interact to generate an outcome given sensory information (Sun, 2016). The diagram below shows the overall structure of Clarion.

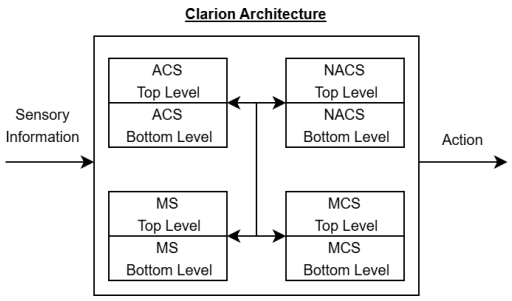


Figure 1: The basic structure of Clarion. Figure recreated from (Sun, 2024a)

Clarion is emphasized in this work because of its two-level structure, which is based on dual-process theory as discussed below.

Dual Process Theory

Dual-process theory of mind was recently popularized by Daniel Kahneman (2011) but had been proposed much earlier by, among others, Reber (1989), Schacter (Schacter, Chiu, & Ochsner, 1993), and Sun (1994). It posits that human thinking is supported by two systems. The first is System 1 (S1) which is used in making often fast, imprecise, unconscious decisions. The second is System 2 (S2), which is used for more deliberate and rational decision-making. These two systems deal with different types of knowledge and processes. S1 works with implicit knowledge/processes, which are less accessible and more holistic, while S2 deals with explicit knowledge/processes, which are more accessible, crisp, and precise (Sun, 2016). The co-existence of these different types of knowledge and processes is well supported by prior empirical work, for example, by Reber (1989), Schacter et al. (1993), and Bornstein and Pittman (1992).

In Sun (1994, 2001), it was shown that neural networks can be a good way to capture implicit knowledge used by S1 and that symbolic representation is better suited for explicit knowledge used by S2. Clarion thus incorporates this theoretical and computational dichotomy in its two-level structure. In humans, these two systems will often interact and benefit one another; similar effects have been demonstrated by Clarion, with demonstrable synergy between its top and bottom levels (Sun et al., 2005).

Bottom-up and Autonomous Learning

Clarion is not unique in incorporating this two-level structure from dual process theory. As discussed by Hélie and Sun (2014), while some other models focus on top-down learning, where a model learns implicit knowledge from the explicit knowledge it possesses or is given, Clarion can perform top-down as well as bottom-up learning. Bottom-up learning is when a model starts with learning implicit knowledge through trial and error and builds up explicit knowledge on that basis through experience (Hélie & Sun, 2014). This is an important aspect of Clarion, as it allows for learning knowledge about the world with no prior knowledge to start with; that is, it allows for autonomous learning (Sun et al., 2001). In humans, some tasks lend themselves better to top-down learning, and others to bottom-up learning, so the inclusion of both in a model is important for psychological realism.

The algorithm that Clarion used to perform bottom-up learning is the Rule-Extraction-Refinement (RER) algorithm (Sun, 2016). Roughly, the basic idea of RER is: if some action decided by the bottom level is successful, then a rule should be extracted. In subsequent interactions, rules are verified and refined using an information gain criterion (Sun & Peterson, 1998). There are three important components of this process: *Extraction, Generalization*, and *Specialization*. Extraction is the first step, which sets up a rule if an action is successful and there is no existing rule matching the current input conditions. Generalization occurs when a rule is successful and it leads to expanding rule conditions, while specialization occurs when a rule fails and it leads to narrowing rule conditions (Sun & Peterson, 1998). If two rules become too similar, they can be merged into one, and rules that consistently fail are deleted (more specific details can be found in Sun (2001, 2016)). The method for rule extraction being developed in this work is a variation of this algorithm adapted to SBERT.

Justifications for Incorporating LLMs

While the current components of Clarion have been extensively validated through rigorous simulations of empirical psychological data, is incorporating LLMs into Clarion justified? The bottom level of Clarion currently utilizes different types of neural networks (in different subsystems) to capture implicit knowledge and processes. This is something that LLMs seem to be able to do well. As argued by Sun (2024a), the knowledge that LLMs store internally can be thought of as mapping roughly to human intuition. Sun (2024a) also ar-

gued that using an LLM at the bottom level of Clarion could be an even better way to represent implicit knowledge and processes than neural networks of the past. Although SBERT (derived from the language model BERT) is not an LLM, it serves as an initial exploration before incorporating large LLMs into Clarion.

Related Work

In terms of integrating LLMs into cognitive architectures, there have been some existing suggestions. In Romero et al. (2023), three methods were suggested: an agency approach where micro agents compete within a cognitive architecture for resources, a modular approach where an LLM could be used to power perceptual or motor control within a cognitive architecture, and the neuro-symbolic approach that utilizes an LLM in the bottom level of Clarion or other similarly structured models. Sun (2024a) also suggested incorporating an LLM in the bottom level of Clarion to perform implicit processing. The method suggested by Sun and the neuro-symbolic approach suggested by Romero et al. are essentially the structure that the present work follows.

A paper that is related to the extraction method described in this paper is Zhu et al. (2024), which uses prompting to extract rules. This model was able to achieve an increase in performance but relies on the large language model's retrieval and specific prompting, which can lead to variance. This paper differs from this method in that the rules are extracted deterministically from the model, and thus should be more reliable and more trustworthy.

Model

This section provides details of the task and the dataset, the model at the bottom level, the process of extracting information from the bottom level to create rules for the top level, and how these two levels combine to produce a prediction.

Task and Dataset

The dataset used to test these ideas was the essays dataset from Pennebaker and King (2000), which contains 2,467 stream-of-consciousness essays labeled with binary values for each of the Big-5 Personality Dimensions.

The Big 5 is the most widely accepted measure of personality developed by Goldberg (1992), and McCrae and Costa (1992). It consists of 5 dimensions: Openness to Experience, Conscientiousness, Extroversion, Agreeableness, and Neuroticism.

Personality prediction has been a surprisingly difficult task, even when utilizing LLMs such as in Killian Jr and Sun (2024) and Yeo, Noh, Jin, and Han (2025). The difficulty of the task and its status as a foundational benchmark for personality detection make it an interesting choice as the first test of the validity of the model presented in this work.

Bottom Level

The first step in training the bottom level of the model is to generate embeddings for each document in the essay dataset. These are computed using the SBERT (Reimers & Gurevych, 2019) model with the "all-mpnet-base-v2" configuration, with an embedding size of 768. These are then used as input into a small three-layer FFN (Feed Forward Neural Network), which generates the personality predictions. In the FFN, three fully connected layers feed into an output layer used for binary predictions of the personality dimensions. A dropout layer is incorporated between the first and second layers to reduce overfitting and is set to shut off 10% of the connections. Figure 2 shows the overall structure of the bottom level.

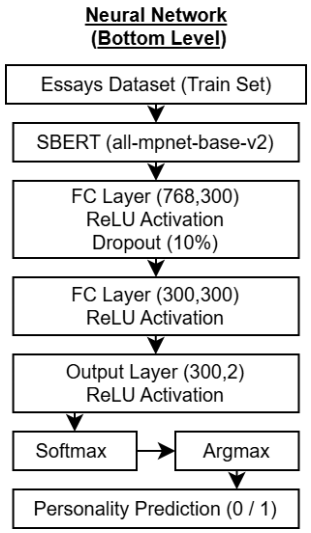


Figure 2: The basic structure of the neural network model at the bottom level of the model

In this paper, only the personality dimension of "openness" is used as an example, but the structure of the model would remain the same for any personality dimension. The loss function used for this model is Cross Entropy (Cox, 1958), and the optimizer is Adam with the default parameters (Kingma & Ba, 2014).

Different versions of the model were trained, some with 80/20 train test splits, with others being 90/10 (this was done to see if the rule set would potentially benefit from more training data). All versions are trained using a batch size of 100 and for 50 epochs. Some details for the performance of the neural network model can be found in the examples section, but most versions hovered around approximately 0.61 F1-score on the openness dimension. There are better models for predicting personality, such as the PADO model proposed by Yeo et al. (2025), but even this model can only manage to reach around 0.7 F1-score on this dimension. This SBERT-based model is used mainly due to its simplicity and ease of use (as a first step in our work), since our goal is to compare the performance of the neural model alone vs. the neural model + rules (regardless of the underlying neural model).

Rule Extraction

The rule extraction process requires that the model at the bottom level be trained first (at least to an extent). The weights are not adjusted after the initial training (for the sake of simplicity, as this current model does not carry out any online learning or top-down learning).

The trained neural network model (with the weights frozen) is used to generate predictions. If the classifier's prediction is correct, then the sliding window process can be carried out to begin extracting rules. This happens separately for the positive and negative correct predictions. Figure 3 shows the overall flow of the rule extraction process.

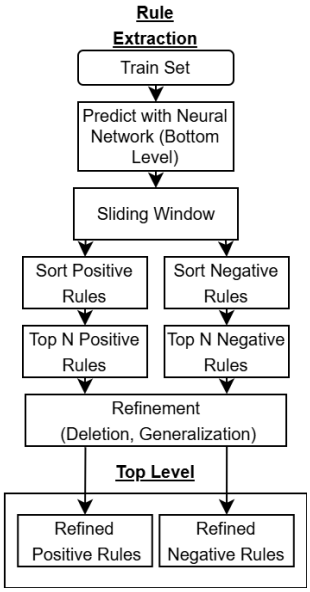


Figure 3: The overall process of extracting and refining rules.

Sliding Window Our rule extraction involves a sliding window (Datar et al. (2002)). For each document that is correctly classified, variations are created, each with a different span of text masked. This is done by replacing words in the document with "[MASK]". The length of these spans varied between 2 - 5 words and they will be referred to as n-gram going forward. This creates a new set of documents including the original document and a set of (# of words in a document - (n - 1)) copies of that document each with a unique span of n words masked. This set of documents is then passed through the neural model, and the loss value is recorded for each. The loss difference (L_d) is then computed for each masked document. This loss difference (where loss is the cross entropy loss function) is defined as:

$$L_d = loss(document_{masked}) - loss(document_{original})$$

The larger the loss difference, the more valuable that span, i.e., the n-gram, is to the model's prediction. The loss difference for each n-gram is calculated, and the top K n-grams are kept and added to a list. Once this process is repeated for every correctly predicted document, this list will now contain the top K n-grams from each document. The last step is simply to sort these and take the top N n-grams and use those as the initial set of rules. This is done for both the positive (1) and the negative (0) predictions, so there will be a set of N positive rules and N negative rules (to be used as an initial rule set). The condition of each of these rules is stored as a pair consisting of an n-gram and its corresponding L_d (n-gram, L_d), as well as in their vectorized form to be able to later compute cosine similarities for refinement and application of rules. The vectorized embedding of an n-gram is calculated using SBERT. This further requires computing embeddings for all possible n-grams for each document to which we wish to apply the rules. Figure 4 presents how the masking of a document is carried out, and how the L_d is calculated.

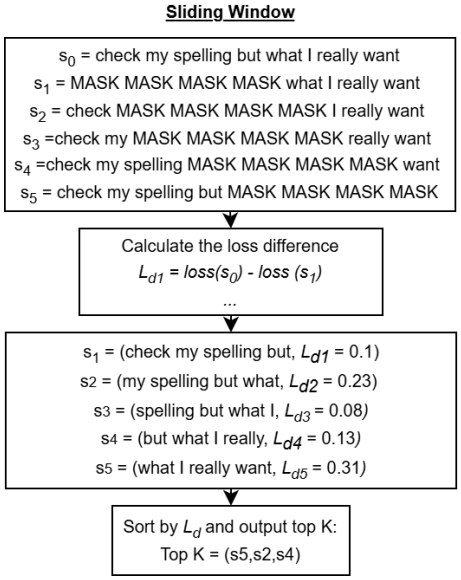


Figure 4: The masking process carried out on a document, and the loss difference calculated for each n-gram.

Once the initial sets of positive and negative rules are obtained, they can then be applied to the dataset. To determine how these rules are applied (before any further refinement), two possible methods are: average similarity and max similarity.

Average Similarity Rule Application This method compares each rule with each n-gram in a document using the cosine similarity function. If this similarity falls above a threshold t_s, then the L_d for that rule is added to a running total. Positive rules add to this total, and negative rules will subtract from this total. After looping through all possible matches of each rule and each n-gram in a document, the values of this running total are used to make a prediction. If the total is positive, output a positive prediction, if negative, output a negative prediction, and if the total is zero, then output no prediction. If the L_d values for the positive and negative values

are imbalanced, then a multiplicative penalty can be applied to the rule set with higher values.

Max Similarity Rule Application This method once again compares each rule with each n-gram in a document using the cosine similarity function. If this similarity is above a threshold t_s then the L_d is added to a list. Separate lists for the positive rules and the negative rules are kept. After checking each rule, the maximum value of each of these lists is compared. If the max value of the positive list is greater than the max value of the negative list, then output a positive prediction. If the reverse is true, then output a negative prediction. Otherwise, output no prediction. If the L_d values for the positive and negative rules are imbalanced, then a bias (offset) term can be included to address this.

After looping through the training dataset once with either of the above methods, steps can then be taken to improve the efficacy of the rules. One method is *deletion* which removes poorly performing rules, and another is *generalization* which expands the scope where a rule is applicable.

Rule Deletion For either method, a record of each time a rule is applied and each time a rule is successful is recorded. This is used to calculate the success rate of each rule s_r:

$$s_r = \frac{\text{\# of times rule was successful}}{\text{\# of times rule was applied}}$$

If this success rate is above a threshold for deletion t_d then the rule is kept, otherwise, the rule is removed from the set.

Rule Generalization Generalization is to determine if a rule with fewer conditions would be better in some way. To do this, a rule is masked in each word position to create n new generalized rules. Those 'child' rules keep the same L_d value as their 'parent' rule when being applied. Figure 5 demonstrates this process.

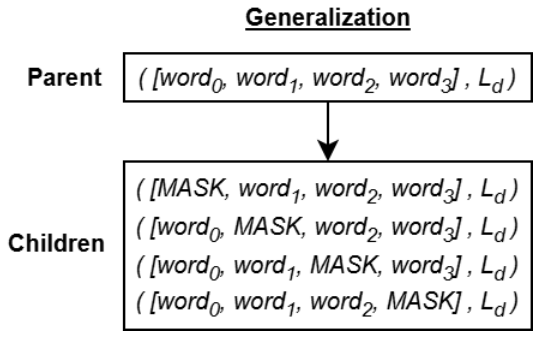


Figure 5: Example of how a 4-gram rule could be generalized

To apply these rules, it requires creating masked versions of the documents as well. So, if using 4-grams, all the 4-grams for that document are calculated, then 4 separate versions of each of those 4-grams are created where the 1st, 2nd, 3rd, and 4th words are masked. This allows for the generalized rules to be able to be matched to those documents. After applying these rules, the deletion threshold can again be applied to determine if these generalized rules are successful. If all child rules fail to be successful, then the original parent rule is added back into the rule set.

Costs of Rule Extraction Each of these methods has an associated cost. As will be explained in the next section on the combination of the top and bottom levels, at least one pass through the training set with the top-level rules is required. Depending on the length of the documents and the number of rules, this can become computationally expensive. Conducting a deletion step requires an additional pass of applying rules to the training set, as does generalization, with doing both requiring two extra passes through the data. These adjustments to the rules can provide benefits (touched upon in the examples section) but do increase the time required to establish a final rule set. Depending on the application, it may be more beneficial to either use deletion, generalization, or just the initially extracted rules.

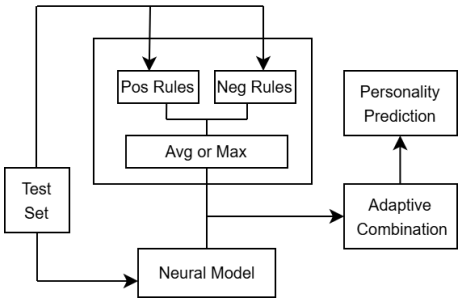


Figure 6: The overall flow of how outputs from the top and bottom levels are combined

Combination of Top and Bottom Level

After establishing a final rule set, it can then be used in combination with the bottom level (or used alone). The combination requires applying the final rule set to the train set and calculating the positive predictive value (PPV), negative predictive value (NPV), and the accuracy that the rule set has on the subset of the data that it applies to. This can be compared to the accuracy, PPV, and NPV of the neural model. If the rules model has better accuracy, then always use its output for predictions when it applies. If just the PPV is superior, then use the rules model only when it outputs a positive prediction. If just the NPV is superior, then use the rules model only when it outputs a negative prediction. If the rules model is not superior in any of these cases, then simply use the neural model for all predictions. Regardless, the rules can always enhance the explainability of the model. All of this is determined on the training set so that when applying the models to the test set, the policy for combining the models is already established.

Some Examples

This section provides a few examples from our preliminary experiments thus far where the combined model provided benefits over using the neural model alone. The examples presented used three different configurations of the model. Configurations one and two use an 80/20 train/test split with different random selections of training and test data, while configuration three uses a 90/10 split.

Example 1: Deletion only (Max), n = 4, K = 3, t_s = 0.9, offset = 0, t_d = 0.65

In this first example, one pass of rule deletion was conducted; the max method was used for similarity matching. Using the above setup of parameters, and configuration two, the accuracy of the model increased from 0.630% (311/494) to 0.638% (315/494). Since the NPV of the top-level rules was greater (0.781 vs 0.747), the negative predictions from the top level were used when they were applicable. The coverage of the top level (percent of cases where rules were applicable) was relatively low at 17%.

Example 2: Deletion only (Avg), n = 4, K = 3, t_s = 0.85, t_d = 0.65

This setup also used rule deletion but with the averaging method for similarity matching. Using this setup of parameters, and configuration one, the accuracy of the model increased from 0.595% (294/494) to 0.603% (298/494). Similarly, this was due to the larger NPV of the top-level on the train set (0.776 vs 0.767), with the coverage of the rules being slightly greater than the previous case at 22%.

```
test set # 391 matched pos rules:
Rule # 20 : (I never read this, 0.1417)
Rule # 78 : (little while every now, 0.0938)
test set # 391 matched neg rules:
Rule # 20 : (many people everywhere I, 0.2494)
total = -0.0139
Predict Negative Openness
```

Figure 7: Output from the top-level rules in Example 2

Example 3: Generalization + Deletion (Avg), n = 5, K = 10, t_s = 0.9, t_d = 0.65

In this example, generalization and deletion were used, with the averaging method for similarity matching. Using this setup of parameters, and configuration three, the accuracy of the model increased from 0.623% (154/247) to 0.632% (156/247). A slightly greater NPV (0.645 vs 0.643) meant that the top level's negative predictions were used when applicable as in both of the prior examples. The coverage, due to generalization, was much greater at 47%.

In general, in the cases where generalization was applied, the coverage was much higher. So adding generalization can help to explain the model's output much more often, even in cases where there is little performance improvement.

```
Positive Rule
(['i', 'listen', 'to', '[MASK]', 'music'], 0.1150)
Matched
like to listen [MASK] music
Predict Positive Openness
```

Figure 8: Output from the top-level rules in Example 3

Limitations and Future Work

The examples above are only illustrations, and far from comprehensive, so more work is needed to verify and improve the results. For the results provided above, although the performance improvement is often relatively small, the added explainability is beneficial regardless.

The limited results may be attributed to the small dataset size, the bottom-level neural network (i.e., SBERT) performance, and so on. Using a larger and more comprehensive dataset to create a more general rule set could be beneficial, and using a large LLM model as the basis could also help to extract better rules. Our plan has been to move on to using real LLMs such as Llama III, Qwen, or Deepseek, at the bottom level of our model, for this dataset and especially other, larger datasets.

Other, alternative methods are also being considered for rule extraction. There are other saliency-based methods such as LIME (Ribeiro, Singh, & Guestrin, 2016), Shapley Values (Shapley, 1951), and integrated gradients (Sundararajan, Taly, & Yan, 2017), which can be applied agnostic of the underlying model, similar to the sliding window n-gram method. There are also attention-specific methods that could be used for rule extraction, such as attention rollout (Abnar & Zuidema, 2020), or the methods proposed by Chefer, Gur, and Wolf (2021a, 2021b). There is also the Neuron-2-Graph method (Foote et al., 2023), which creates an interpretable graph from neural networks. Some of these methods may be able to extract more useful or more complex rules than the sliding window n-gram method.

Conclusion

The present paper describes our ongoing work on rule extraction within the two-level hybrid structure of Clarion, as applied to the personality prediction task. SBERT was used as the basis of the neural model at the bottom level, and the top-level n-gram rules model was constructed through bottom-up learning using a combination of sliding window n-gram extraction, deletion, and generalization. There are cases where the combination of the top and the bottom level leads to enhanced performance and explainability, but much further work is required to validate the method. Along with tweaking parameters and generating more results on the essay dataset, further explorations with various LLMs and various rule extraction methods and algorithms are to be conducted going forward.

Acknowledgments

Thanks to Dr. James Pennebaker for generously providing the essays dataset that was used to test our models, Dr. Tomek Strzalkowski for providing computational resources, and Abraham Sanders for his suggestion of saliency-based methods for rule extraction.

References

Abnar, S., & Zuidema, W. (2020). Quantifying attention flow in transformers. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 4190–4197). doi: 10.18653/v1/2020.acl-main.385

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. doi: 10.1207/s15327051hci1204$_5$

Bornstein, R. F., & Pittman, T. S. (Eds.). (1992). *Perception without awareness: Cognitive, clinical, and social perspectives*. Guilford Press.

Chefer, H., Gur, S., & Wolf, L. (2021a). Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In *Proceedings of the ieee/cvf international conference on computer vision (iccv)* (pp. 387–396). doi: 10.1109/ICCV48922.2021.00045

Chefer, H., Gur, S., & Wolf, L. (2021b). Transformer interpretability beyond attention visualization. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr)* (pp. 782–791). doi: 10.1109/CVPR46437.2021.00084

Costa, P. T., & McCrae, R. R. (1992). The five-factor model of personality and its relevance to personality disorders. *Journal of Personality Disorders*, *6*(4), 343–359. doi: 10.1521/pedi.1992.6.4.343

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, *20*(2), 215–232. doi: https://doi.org/10.1111/j.2517-6161.1958.tb00292.x

Datar, M., Gionis, A., Indyk, P., & Motwani, R. (2002). Maintaining stream statistics over sliding windows. In *Proceedings of the thirteenth annual acm-siam symposium on discrete algorithms* (pp. 635–644).

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., ... many others (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*. doi: 10.48550/arXiv.2501.12948

Foote, A., Nanda, N., Kran, E., Konstas, I., Cohen, S., & Barez, F. (2023). Neuron to graph: Interpreting language model neurons at scale. In *arxiv.* doi: https://doi.org/10.48550/arXiv.2305.19911

Goldberg, L. R. (1992). The development of markers for the big-five factor structure. *Psychological Assessment*, *4*(1), 26–42.

Hélie, S., & Sun, R. (2014). Autonomous learning in psychologically-oriented cognitive architectures: A survey. *New Ideas in Psychology*. (In press) doi: https://doi.org/10.1016/j.newideapsych.2014.03.002

Kahneman, D. (2011). *Thinking, fast and slow*. Farrar, Straus and Giroux.

Killian Jr, J., & Sun, R. (2024). Detecting big-5 personality dimensions from text based on large language models. In *Delta 2024, ccis 2172* (pp. 264–278). Springer Nature Switzerland AG. doi: 10.1007/978-3-031-66705-3$_1$8

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. doi: https://doi.org/10.48550/arXiv.1412.6980

Laird, J. E. (2019). *The soar cognitive architecture*. Cambridge, MA: MIT Press.

Pennebaker, J., & King, L. (2000, 01). Linguistic styles: Language use as an individual difference. *Journal of personality and social psychology*, *77*, 1296-312. doi: 10.1037//0022-3514.77.6.1296

Reber, A. S. (1989). Implicit learning and tacit knowledge. *Journal of Experimental Psychology: General*, *118*(3), 219–235. doi: 10.1037/0096-3445.118.3.219

Reimers, N., & Gurevych, I. (2019, November). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp).* Hong Kong, China: Association for Computational Linguistics. doi: 10.18653/v1/D19-1410

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144. doi: 10.1145/2939672.2939778

Romero, O. J., Zimmerman, J., Steinfeld, A., & Tomasic, A. (2023). Synergistic integration of large language models and cognitive architectures for robust ai: An exploratory analysis. *arXiv preprint arXiv:2308.09830*. doi: 10.48550/arXiv.2308.09830

Schacter, D. L., Chiu, C. Y. P., & Ochsner, K. N. (1993). Implicit memory: A selective review. *Annual Review of Neuroscience*, *16*, 159-182. doi: 10.1146/annurev.ne.16.030193.001111

Shapley, L. S. (1951). *Notes on the n-person game – ii: The value of an n-person game* (Research Memorandum No. RM-670). Santa Monica, CA: RAND Corporation.

Sun, R. (1994). *Integrating rules and connectionism for robust commonsense reasoning*. USA: John Wiley & Sons, Inc.

Sun, R. (2001). *Duality of the mind: A bottom-up approach toward cognition*. New York: Psychology Press. doi: https://doi.org/10.4324/9781410604378

Sun, R. (Ed.). (2008). *The cambridge handbook of computational psychology*. Cambridge University Press. doi: 10.1017/CBO9780511816772

Sun, R. (2016). *Anatomy of the mind: Exploring psy-*

chological mechanisms and processes with the clarion cognitive architecture. Oxford University Press. doi: 10.1093/acprof:oso/9780199794553.001.0001

Sun, R. (2024a). Can a cognitive architecture fundamentally enhance llms? or vice versa? *arXiv preprint arXiv:2401.10444*. doi: https://doi.org/10.48550/arXiv.2401.10444

Sun, R. (2024b). Dual-process theories, cognitive architectures, and hybrid neural-symbolic models. *Neurosymbolic Artificial Intelligence*, 1-9. doi: 10.3233/NAI-240720

Sun, R., Merrill, E., & Peterson, T. (2001). From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Science*, *25*(2), 203–244. doi: 10.1207/s15516709cog2502$_2$

Sun, R., & Peterson, T. (1998). Autonomous learning of sequential tasks: Experiments and analyses. *IEEE Transactions on Neural Networks*, *9*(6), 1217–1234. doi: 10.1109/72.728364

Sun, R., Slusarz, P., & Terry, C. (2005). The interaction of the explicit and the implicit in skill learning: A dual-process approach. *Psychological Review*, *112*(1), 159–192. doi: 10.1037/0033-295X.112.1.159

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th international conference on machine learning (icml)* (pp. 3319–3328). doi: https://doi.org/10.48550/arXiv.1703.01365

Yeo, H., Noh, T., Jin, S., & Han, K. (2025, January). Pado: Personality-induced multi-agents for detecting ocean in human-generated texts. In *Proceedings of the 31st international conference on computational linguistics* (pp. 5719–5736). Abu Dhabi, UAE: Association for Computational Linguistics.

Zhao, M., Xu, D., & Gao, T. (2024). From cognition to computation: A comparative review of human attention and transformer architectures. *arXiv preprint arXiv:2407.01548*. doi: 10.48550/arXiv.2407.01548

Zhu, Z., Xue, Y., Chen, X., Zhou, D., Tang, J., Schuurmans, D., & Dai, H. (2024). Large language models can learn rules. *arXiv preprint arXiv:2310.07064*. doi: https://doi.org/10.48550/arXiv.2310.07064

Model Predictions and Implications for Chasing Subtlety

Maria Kon (maria.r.kon.civ@us.navy.mil)[1,2]
Sangeet Khemlani (sangeet.s.khemlani.civ@us.navy.mil)[1]
Gregory Francis (gfrancis@purdue.edu)[1]
Andrew Lovett (andrew.m.lovett.civ@us.navy.mil)[1]
[1]Navy Center for Applied Research in Artificial Intelligence, U.S. Naval Research Laboratory, Washington, DC 20375 USA
[2]Department of Psychological Sciences, Purdue University, 703 Third Street, West Lafayette, IN 47907 USA

Abstract

Chase detection involves tracking objects and comparing their locations over time. What is it about the relative spatial relations of two objects that helps you perceive one as chasing the other rather than, say, merely moving in the general direction of the other object? A recent model of chase detection provided an explanation in terms of an attentional strategy. However, it is unclear if this model generalizes or has predictive power since it was fit to experimental data. Here we examine whether this model explanation extends to and predicts a frequently studied chasing cue: chasing subtlety—the degree to which the chaser deviates from the most direct path to its target. To test the model, we made preregistered model predictions from simulations run *prior* to data collection. We then conducted two experiments where chasing subtlety varied. Overall, the model did a good job predicting response time and accuracy patterns across most conditions. Additionally, it predicted specific videos that had the highest error rates. Thus, we show that the model explanation extends to chasing subtlety and, more broadly, that the model can be used to generate a falsifiable theory of chase detection.

Keywords: chase detection; chasing; relations; dynamic scenes

Introduction

Humans possess an impressive capacity for understanding complex visual scenes, including identifying the actions of agents and the intentions behind those actions. One task developed to study this understanding is *chase detection*, in which an individual observes objects moving through a scene and determines whether one object is chasing another. Chase detection requires tracking objects and comparing their locations and motion patterns over time to identify subtle differences between, e.g.,"following", where the intention is to maintain some distance between agents, and "pursuing", where the intention is to close the gap. Many questions remain about how people infer the correct intention.

Recent work has investigated possible cues for determining whether agents are involved in a chase, including distance between the chaser and its target (Meyerhoff, Schwan, & Huff, 2014a,b), the direction that the chaser faces (Gao, McCarthy, & Scholl, 2010; Gao, Newman, & Scholl, 2009), the number of objects in a scene (Gao et al., 2019; Kon et al., 2024; Meyerhoff et al., 2013), and *chasing subtlety*, the degree to which the chaser can deviate from the most direct path to its target (Gao et al., 2019; Gao et al., 2009; Meyerhoff et al., 2013).

We recently developed a cognitive model for the chase detection task, to aid in exploring problem-solving strategies and stimulus factors that may influence performance (Kon, Khemlani & Lovett, 2024). The core claim of the model was that chase detection, like other demanding visual tasks, depends on strategically projecting spatial attention onto a visual scene. To determine whether one object is in pursuit of another, the model tracks the object, determines its motion trajectory, and then projects spatial attention along that trajectory to identify another object that may be a potential target of pursuit. This process is engaged repeatedly, and if a prospective pursuer is consistently moving towards the same prospective target, then it is likely that a chase is occurring.

We evaluated our model on a novel chase detection task, in which the possible pursuer was always a red circle and the possible targets were circles of unique colors. This color-coding simplified detection and tracking in a way that isolated specific factors that contribute to chase detection. The task utilized a two-stage design in which participants first pressed a button to indicate whether a chase was occurring (Stage 1) and then indicated the circle being chased (Stage 2), allowing for measures of response time and accuracy. After gathering human data on the task, we presented the same trial videos to the model and found an overall close fit to the human data.

Although the modeling results were promising, they suffered from several limitations. 1) The human data was gathered first, and free parameters in the model were selected to maximize the fit to this data. Additionally, there was only a small set (20) of stimulus videos. Thus, it remains unclear how well the model will generalize to other situations. 2) The study looked at the core chase detection task and varied only set size, without considering other factors that are believed to contribute to detection performance.

To overcome these limitations, we present a new pair of studies with the following changes. 1) Before the study was given to human participants, the model was run on the study with specific parameter values, and the resulting model predictions were preregistered on OSF. 2) A larger set of videos (180) was used. In these videos, we varied chasing subtlety, the degree to which the pursuing red circle moves directly towards the target, to explore how this factor affects the model's performance and fit to the human data. While there are other studies that measure the effect of chasing subtlety on speed and accuracy of chase detection (Gao et al., 2019; Gao et al., 2009; Meyerhoff et al., 2013), these studies used small sample sizes, ranging from 12-22. So, we also wanted to see whether some of these results are replicable with a larger sample size within this experimental paradigm.

Experiment Design

As in Kon et al. (2024), the present experiment used a two-stage chase detection task (Figure 1).

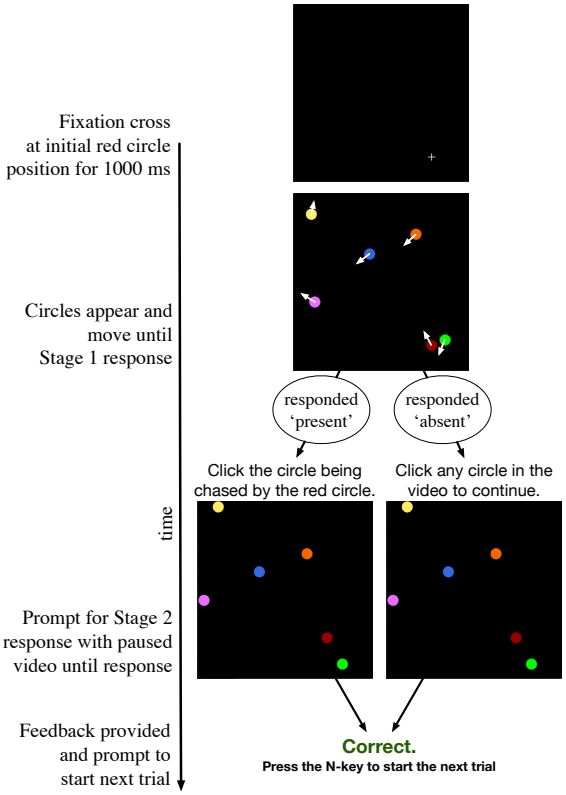


Figure 1: Schematization of an experimental trial in Experiment 1. After seeing a fixation cross at the initial position of the red circle, participants saw video of moving circles (motion is represented in this figure by the white arrows). When participants indicated whether a chase is present or absent, the video paused and a prompt to click on a circle appeared. Feedback was provided after each trial.

Each trial began with a fixation cross at the initial location of the red circle. Next, the red circle appeared with either one other circle (set size 2, including the red circle) or five other circles (set size 6). The initial location and initial trajectory of each circle was randomized with the constraint that no circles overlapped. All circles traveled at the same fixed speed. The red circle moved along a straight path toward the center of the chased circle with its direction updated every 50 ms. The circles moved until participants responded by pressing a key to indicate whether a chase was present or absent (Stage 1), or until the video ended after 22 seconds. Next, if participants indicated there was a chase present, they clicked on the circle they believed was being chased (Stage 2).

The trials in which the red circle was engaged in a chase (*chase-present* trials) varied in their chasing subtlety, which is the extent to which the chaser moves directly towards the target. We followed Gao et al. (2009) and Meyerhoff et al. (2013) by operationalizing subtlety as the difference in degrees between the direction the chaser travels and the direct path (0°) to the target. For 0° chase-present trials, the red circle updated its position towards the center of the target, whereas for 30° chase-present trials, it could deviate anywhere from -30° to 30° from that path.

For the study, four factors were varied: set size (2 or 6), chasing condition (present or absent), chasing subtlety (0°, 30° or 60°), and target color (one of five possible colors). We provide an example video for each condition here: osf.io/8cefh/. Three videos were generated for each combination of conditions, yielding 180 total videos. Chase-absent trials were generated exactly the same as chase-present trials, except that the target was an invisible circle. Thus, the red circle's motion patterns did not differ depending on whether it was chasing a visible circle or not.

The Model and Predictions

We developed a computational model of chase detection (see Kon et al., 2024, for details) based on the idea that an observer directs spatial attention to task relevant locations (Posner, 1980; Ullman, 1984). The model evaluates whether a red circle is chasing another by (1) finding and tracking the red circle over time to estimate its future trajectory based on past motion; (2) scanning along that trajectory; and (3) identifying what, if any, circle lays within a scan window traversing the trajectory. The more times a particular object is found along the expected path of the red circle, the more evidence the red circle is chasing this object.

The model is built within the ARCADIA framework, which was designed to explore attention's role in perception, cognition, and action (Bridewell & Bello, 2016). ARCADIA models are implemented as a set of components that process information and generate output; and an attentional strategy is used to select one piece of output as the focus of attention, which drives further processing. To these components, the chase detection model adds a set of stopping rules for determining when sufficient evidence has accumulated before producing a response on a trial.

Each ARCADIA processing cycle works as follows. On each cycle, components have access to: the output from all components on the previous cycle, a single output element that was selected as the focus of attention on the previous cycle, and data generated by sensors—typically, this grabs frames from a video, such as the chase detection stimuli. Components are essentially functions that process this information to generate new output elements that will be available on the following cycle. Depending on their purpose, different components will respond to different information.

The chase detection model relies on five key components. The **image segmenter** processes the images coming from the sensor, performing figure-ground segmentation (Palmer & Rock, 1994) to pick out segments corresponding to the colored circles in the video. These segments are used by the remainder of the model (Figure 2). The **color highlighter** identifies each segment's color and puts it forward as a candidate for attention. In the current model, the attentional strategy prioritizes attending to the red circle since it is always the potential pursuer. While attention is focused on the red circle, its positional information is used to calculate its motion trajectory (Figure 2, cyan line). After this trajectory information

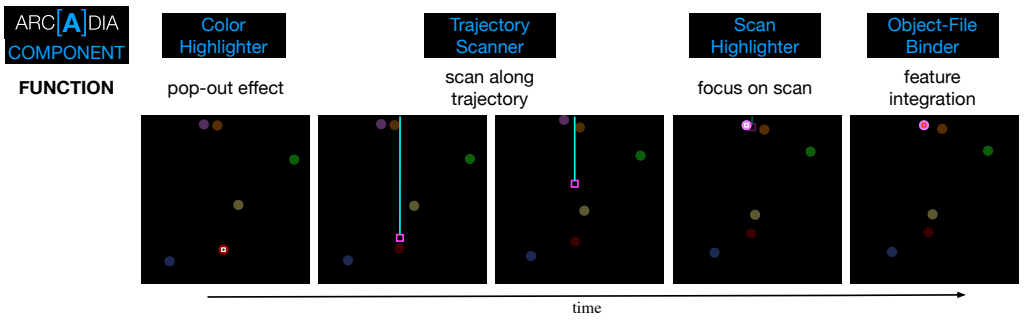


Figure 2: Example illustrating the function of each model component. Areas not grayed out indicate the focus of attention at a given time.

is available, the **trajectory scanner** projects a small window of spatial attention along that trajectory (the *scan window*, represented by the magenta box in Figure 2; see Gerstenberg et al., 2017). If that scan window intersects another circle, the **scan highlighter** highlights that circle as a candidate for attention (Bello et al., 2018). Finally, the **object file binder** generates a representation of this object (Treisman & Gelade, 1980), and its intersection count is incremented.

The model's stopping rules codify different strategies for determining when to respond on the chase detection task. The strategies can be varied based on four free parameters to the model. The stopping rules are as follows. 1) If the intersection count for a circle reaches the *intersection counter threshold*, then the model responds that this circle is being chased by the red circle. 2) If no intersection count reaches this threshold but *maximum time* is reached, then the model responds that it detects no chase. 3) If at *initial time check* no intersection count is at least at the value of the *initial intersection threshold*, then the model responds that there is no chasing detected. This third stopping rule can generate a quick response when there is minimal evidence of chasing.

A fifth free parameter is *scan window size* (diameter of the scan window). Our prior modeling work suggested that varying it could explain how differences in set size affect performance. In this study, there were either 2 or 6 objects in each video. We got the best fit to human data when the model used a small scan window with a large set size (6), and large scan window with a small set size (2). Intuitively, this makes sense: when the scene is cluttered with a large number of objects, one would want to focus in on a small area to avoid being distracted by irrelevant objects. But when there is only a single candidate for the target, a large scan window can maximize the chances of hitting that circle when scanning.

Model Results and Predictions

We ran several sets of simulations where we varied scan window size (with widths 71-182% of a circle diameter, in increments of approximately 12%), intersection counter threshold (4-11 in integer increments) and maximum time values (7-21 seconds, in increments of 2), while we fixed the initial time check (6 seconds) and initial intersection threshold (1) across simulations. The full results of these simulations and detailed discussion of them are beyond the scope of this paper; however, they are provided in the preregistration (osf.io/59gkc/).

Based on the assumption that a human observer should make fast yet accurate responses, we made predictions about performance on this task based on the results of parameter values that produce both low response times and low error rates across all conditions. In other words, rather than fitting the model to data, we made predictions about performance prior to collecting data by examining model performance for parameter values that resulted in both the lowest mean response times and the highest mean accuracy rate across conditions.

In line with our past findings, such performance with low response times and high accuracy occurs for set size 2 when larger scan window sizes are used and for set size 6 with smaller scan window sizes. We also identified a subset of parameters that reflected a speed-accuracy tradeoff. It is characterized by larger scan window sizes for set size 2, smaller scan window sizes for set size 6, an intersection counter threshold of 5 or 6, and a maximum time from 15-17 seconds.

Figure 3A shows representative model results given these parameters. The several qualitative predictions about human performance based on model results are summarized below.

1. On chase-present trials (blue lines in Figure 3), response times will increase with chasing subtlety.
2. On chase-absent trials (red lines), response times will be unaffected by chasing subtlety.
3. All response times will increase with set size.
4. On chase-present trials, Stage 1 accuracy will be near ceiling for 0° and 30° subtlety, but will decrease for 60°.
5. On chase-present trials at 60°, performance will be lower for set size 2 than set size 6. This surprising prediction is the only case where set size 6 is easier.

The model also identifies particular videos that may result in lower accuracy and predicts the response times for these videos and which incorrect target circle will tend to be chosen. These are videos for which the model has a high error rate even when it uses parameter values that resulted in overall low speed and high accuracy. We discuss some of these videos, which are in our preregistration, in the *Error-Prone Videos* section below.

Experiment 1

To test model predictions about human performance on a larger stimulus set, we conducted an experiment after preregistering the predictions, using the same task and stimulus set

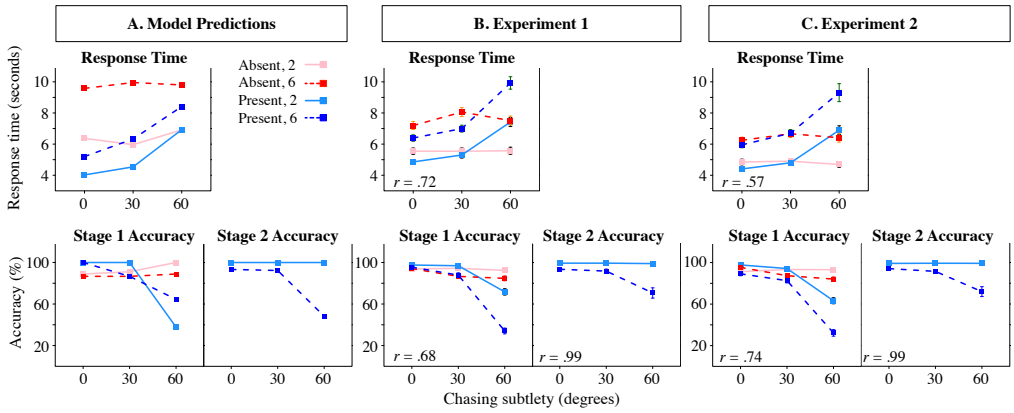


Figure 3: Model predictions and experimental results, with correlations between model results and data for each experiment. Mean response times were calculated from correct trials only. Error bars represent one standard deviation of the mean.

used in model simulations. Various chasing subtleties were used in the stimuli to investigate whether the model strategy generalizes to this frequently studied cue (Gao et al., 2019; Gao et al., 2009; Meyerhoff et al., 2013) and whether some of the key results of past chasing subtlety studies replicate with a larger sample.

Method

Participants 102 participants (mean age = 42.05 years; 56 females, 44 males, 1 other, 1 prefer not to answer) completed the study on the Amazon Mechanical Turk online platform in exchange for US$3.75. We excluded data from 5 participants with mean accuracy $<$ 2 standard deviations from pooled mean accuracy (82%). All but 3 participants reported normal color vision, with 2 reporting suspected colorblindness and 1 who did not respond; however, each yielded 82%, 81% or 89% accurate responses, so we retained their data. We analyzed the remaining data from $n = 97$ participants.

Materials, Procedure and Design The study used custom JavaScript and HTML code within the nodus-ponens package (Khemlani, 2022). After navigating to the webpage with the study, participants saw instructions that described the task, informed them that the red circle was always the potential chaser, stated the purpose of the fixation cross, and presented a few example videos and interactive example trial. Participants initiated each trial by pressing a key, which resulted in a video playing that showed a fixation cross followed by moving circles. Participants responded with the *F*- and *J*-keys on their keyboard to indicate whether there was a chase present or not (Stage 1 response), with key assignment counterbalanced across participants. The key press paused the video, and participants saw a prompt to click one of the circles (Stage 2 response) using their mouse. Specific feedback was provided after each trial, i.e., "Correct."; "Incorrect – red was chasing"; "Incorrect – no chasing"; "Incorrect circle", in addition to a warning if the Stage 1 response was too fast ($<$ 500 ms) or too slow ($>$ 20 s).

Each participant saw 4 practice trials with chasing subtlety 0°, which was followed by 32 experimental trials in randomized order, yielding a 2 (chase present vs. absent) × 2 (2 circles or 6 circles) × 3 (chasing subtlety 0°, 30° or 60°) repeated-measures design with these 12 total conditions repeated 3 times. For each trial a video was chosen randomly from the subset of videos in the pool of 180 videos that had a specified set of the independent variables of interest, e.g., chase-present, set size 6, chasing subtlety 30°, with a randomized target color and version number.

Results and Discussion

Figure 3B shows the mean response times and Stage 1 and Stage 2 accuracies for each condition. Analyses for response times were run on correct trials only. We fit the data to a linear mixed model using the nlme package (Pinheiro et al., 2021) in R (version 4.3.1; R Core Team, 2023). We found a significant effect of set size ($\chi^2(2) = 401.63$, $p < .001$) and of chasing subtlety ($\chi^2(2) = 117.16$, $p < .001$) on response time, but the presence of a chase did not reliably impact response times ($\chi^2(2) = 0.14$, $p = .707$). Pairwise contrasts with Tukey adjustment indicated that response time significantly increased with chasing subtlety (0° v. 30°: $b = 440.63$, $t(2799) = 4.48$, $p < .001$; 0° v. 60°: $b = 1181.27$, $t(2799) = 11.06$, $p < .001$, 30° v. 60°: $b = 741.27$, $t(2799) = 6.89$, $p < .001$) and with set size ($b = 1764.14$, $t(2799) = 20.84$, $p < .001$).

The pattern of response time data (Figure 3B, top plot) generally matches the pattern predicted by the model (Figure 3A, top plot; $r = .72$). As predicted, when there is a chase, response times for set size 2 are lower than for set size 6, and response times increase with chasing subtlety. And, as predicted, response times are relatively flat across chasing subtlety for chase-absent trials. However, the model performs worse than humans for these trials; people are much faster to indicate correctly that there is no chase.

How does our data compare to other studies? For chase-present conditions, only Gao et al. (2019) measured responses across chasing subtlety conditions, and the results from Experiment 1 generally matches: lower response times for smaller set sizes with a large increase in response time at 60°. No study provides chase-absent trial response times.

Full analyses of Stage 1 and Stage 2 accuracy are available

at osf.io/8cefh; here we compare qualitative results and model predictions against the results of other studies. For Stage 1 accuracy (Figure 3B), chase-present performance is near ceiling except for 60°, and accuracy is lower for set size 6 than set size 2. Although the pattern for 0° and 30° resembles that of the model (Figure 3A), the model's prediction for 60° (performance should be worse for set size 2 than for 6) does not match the data, resulting in a lower correlation ($r = .68$). Only Gao et al. (2009) measure Stage 1 accuracy (detection accuracy) and for set size 4 chase-present trials only, making it difficult to directly compare our Stage 1 results with existing studies.

Stage 2 accuracy (Figure 3B) strongly resembles ($r = .99$) the model's predictions (Figure 3A), with set size 2 performance at ceiling. There is also a drop-off in accuracy for set size 6 with chasing subtlety of 60°, although people are more accurate than the model for this condition. Additionally, the pattern for set size 6 mirrors the results for accuracy reported by Gao et al. (2009, 2019) and Meyerhoff et al. (2013).

Experiment 2

Experiment 1 gave feedback on each trial. We were curious about the role of feedback on performance for two related reasons. First, the model does not receive feedback. So, we wanted to ascertain whether a version of the experiment without feedback would better approximate model results, particularly for the 60° subtlety condition. Second, it may be argued that chases with high chasing subtleties are not really chases at all. Although they may be defined in the generation of stimuli as a chase, they might not be considered chases by participants. Receiving feedback on each trial may cause participants to learn to classify videos with higher chasing subtleties as chases, even though outside of the experiment, these videos would not be regarded as chases. If this is the case, we would expect accuracy for 60° chase-present trials to be much lower when no feedback is given. To examine the impact of feedback on performance, we conducted a second experiment with the same design as Experiment 1 except that feedback did not follow experimental trials.

Method

Participants 102 naïve participants (mean age = 19.54 years; 65 females, 36 males, 1 other) were recruited from Purdue University in exchange for course credit. Data from 4 participants with mean accuracy less than 2 standard deviations from the mean (81%) were excluded. Two participants did not have self-reported normal color vision, and the data from one of these participants was among the 4 excluded for having a low accuracy rate. We retained the data from the other (92% accurate). We also excluded data from: 3 participants who chose not to report their age, 1 who reported being younger than 18, and 7 due to technical issues. We analyzed the remaining data from $n = 87$ participants.

Materials, Procedure and Design This experiment was identical to Experiment 1 except for the following. Participants received no feedback after each experimental trial. Since we had more time with this subject pool, each participant saw more experimental trials: we repeated the 12 total conditions (2 (chase present vs. absent) × 2 (2 circles or 6 circles) × 3 (chasing subtlety 0°, 30° or 60°)) 5 times resulting in 60 randomly interleaved experimental trials per participant.

Results and Discussion

Experiment 2 yielded results similar to those of Experiment 1 (see Figure 3C) with a significant effect of set size ($\chi^2(2) = 472.72$, $p < .001$) and chasing subtlety ($\chi^2(2) = 120.16$, $p < .001$) on response time, but, unlike Experiment 1, the impact on response time of whether a chase is present was also significant ($\chi^2(2) = 29.70$, $p < .001$). Pairwise contrasts with Tukey adjustment indicated that response time significantly increased with chasing subtlety (0° v. 30°: $b = 387$, $t(4157) = 4.64$, $p < .001$; 0° v. 60°: $b = 1008$, $t(4157) = 11.03$, $p < .001$, 30° v. 60°: $b = 741.27$, $t(4157) = 6.89$, $p < .001$), set size ($b = 621$, $t(4157) = 22.36$, $p < .001$), and when a chase was present rather than absent ($b = 399$, $t(4157) = 5.46$, $p < .001$).

Thus, we have replicated the results from Experiment 1 from a different population, providing stronger support for the conclusions drawn from Experiment 1. Additionally, it seems that feedback does not have much impact on accuracy.

Error-Prone Videos

The model makes specific predictions about which videos are likely to receive incorrect responses. We focus on the three videos with 0° chasing subtlety for which the model, with parameters values that led to good overall performance, had a 100% error rate. Our preregistration provides details about these parameter values and how we made these predictions. Each of these videos can be found at osf.io/8cefh. Because the results from Experiments 1 and 2 were similar, we pooled the responses from these experiments to identify videos with high error rates.

Figure 4 compares model performance with empirical results for each of these videos. Video 1 is a chase-absent video with a red circle and a green circle, with the red moving towards the green circle at times during the first half of the video. The model predicted that people would tend to incorrectly respond that the red circle is chasing the green circle, with a response time in the range of 7.42-9.70 seconds. The gray bar of Figure 4A, left plot, shows the percent of incorrect Stage 2 responses expected for each color. So, the model predicts that 100% of the incorrect responses will indicate that the green circle was being chased. In line with the model's prediction, the same video was the most difficult at Stage 1 for humans across all 0° subtlety videos. Humans had a Stage 1 error rate of 34% on this video (shown in the title of Figure 4A, right plot). Additionally, all participants who responded incorrectly indicated that the green circle was being chased, with a mean response time of 7.92 seconds (Figure 4A, right plot).

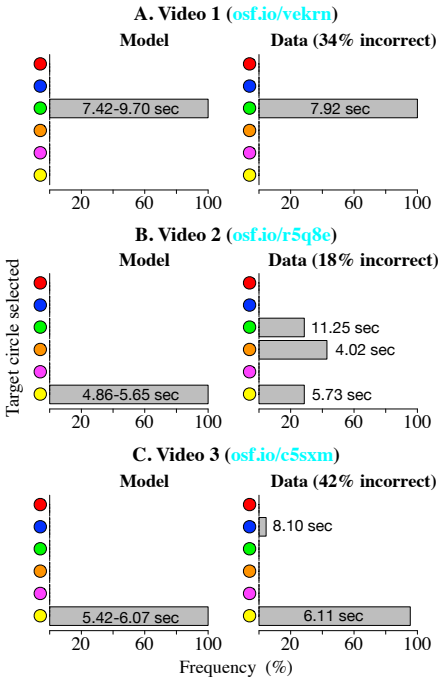


Figure 4: Left column: Model predictions for three videos with the highest simulated error rates, which show the response time associated with an incorrect response and what color will be chosen. Right column: Data for incorrect responses for these videos, indicating how frequently a particular color was chosen and the mean Stage 1 response time for that color. 'XX% incorrect' in the titles indicates the percent of video views with an incorrect response. Each plot shows information for incorrect responses for a particular video, i.e., what circle was thought to be chased (each bar indicates the percent of incorrect responses (x-axis) where a particular circle (y-axis) was selected) and the mean response time for the Stage 1 response on these trials (indicated by the number on/beside a bar).

Video 2 is a chase-absent video with set size 6. At different points in the video, the red circle moves towards a subset of the circles for a short time. The model predicts that people should tend to respond incorrectly that the yellow circle is being chased, with a response time in the range of 4.86-5.65 seconds (Figure 4B, left plot). For humans, however, this video did not rank among the top error-prone 0° chasing subtlety videos, with only 18% of participants responding incorrectly. Further, those that did respond incorrectly were not unanimous in indicating that the yellow circle was being chased (Figure 4B, right plot), yet their mean response time (5.73 sec) falls near the range predicted by the model.

Video 3 is a set size 6 chase-present trial where the orange circle is chased. However, the yellow circle seems to follow the orange circle for a short time at the beginning of the video. The model predicts the people will tend to provide a correct Stage 1 Response and, thus, respond that there is a chase around 5.42-6.07 seconds. However, that model also predicts that people will tend to make a Stage 2 error, indicating that yellow, rather than orange, is the circle being chased (Figure 4C, left plot). In line with the prediction, this video had the highest Stage 2 error rate (42%) among those with a 0° chasing subtlety, with the vast majority indicating that the yellow circle was being chased at around 6.11 seconds.

Overall, the model predicted the 0° chasing subtlety video with the highest Stage 1 error rate (Video 1), and the video with the highest Stage 2 error rate (Video 3). Additionally, in these cases, the model made accurate predictions about when people who made these errors tended to respond and what circle they tended to select. However, the model was not a good predictor regarding Video 2. Most people seemed to wait long enough to correctly identify there was no chase.

General Discussion

Previous research has identified a number of factors that influence chase detection in humans. The present work builds on that body of research while focusing on one key factor, chasing subtlety. Compared to prior studies, the present study uses larger sample sizes and provides an explanation, via a computational model, for why performance degrades when a pursuing object does not move directly towards its target.

More broadly, the present work explains chase detection by combining a) a model, based on the claim that participants direct spatial attention towards potential chase targets to perform the task, and b) a strategy for assigning values to the model's free parameters before human data has been gathered, based on finding a balance between low response times and high accuracy across conditions. We found the model parameter values that best achieved this balance result in a model that focuses attention on a narrow area when the scene is more crowded and a larger area when there are few objects. This provides support for and an explanation of our claim that the size of the focus of attention differs depending on the number of distractors in a scene (Kon et al., 2024): a narrow focus (wider focus) when there are many (few) objects tends to lead to lower response times but also higher accuracy. The result is a model capable of both predicting overall human performance and identifying videos that will be especially difficult for humans.

While the pattern of model results generally matches empirical results, humans outperform the model when there is no chase, responding faster across all set sizes and chasing subtlety conditions. This implies that the model stopping rules do not reflect those used by humans on this task. Future work should systematically survey plausible alternative cognitive stopping rules. For example, an earlier initial check may be afforded by having a dynamic scan window that gets larger as it moves away from the potential chaser.

We have demonstrated that this computational model can be used to make empirically testable predictions about human performance on chase detection tasks *prior* to data collection. This shows that the model allows us to develop a falsifiable theory of chase detection, one that we intend to expand and refine through future model development and empirical tests.

Acknowledgments

This work was supported by a National Research Council Associateship Award to MK and funding from the Naval Research Laboratory to SK. Data for Experiment 1 were collected by Knexus Research Corporation. The views expressed in this paper are solely the authors' and should not be taken to reflect any official policy or position of the United States Government or the Department of Defense.

References

Bello, P., Lovett, A. M., Briggs, G., & O'Neill, K. (2018). An Attention-Driven Computational Model of Human Causal Reasoning. In *CogSci*.

Bridewell, W., & Bello, P. F. (2016). A theory of attention for cognitive systems. In *Proceedings of the 4th Annual Conference on Advances in Cognitive Systems* (pp. 1–16). Evanston. Conference on Advances in Cognitive Systems (pp. 1–16). Evanston.

Gao, T., Baker, C. L., Tang, N., Xu, H., & Tenenbaum, J. B. (2019). The cognitive architecture of perceived animacy: Intention, attention, and memory. *Cognitive Science*, *43*(8), e12775.

Gao, T., McCarthy, G., & Scholl, B. J. (2010). The wolfpack effect: Perception of Animacy irresistibly influences interactive behavior. *Psychological Science*, *21*(12), 1845–1853.

Gao, T., Newman, G. E., & Scholl, B. J. (2009). The psychophysics of chasing: A case study in the perception of animacy. *Cognitive Psychology*, *59*, 154–179.

Gerstenberg, T., Peterson, M. F., Goodman, N. D., Lagnado, D. A., & Tenenbaum, J. B. (2017). Eye-tracking causality. *Psychological Science*, *28*(12), 1731–1744.

Khemlani, S. (2022). nodus-ponens: A light, full-stack framework for running high-level reasoning and cognitive science experiments in Node.js. [Computer software]. Retrieved from https://www.npmjs.com/package/nodus-ponens

Kon, M., Khemlani, S., & Lovett, A. (2024). Measuring and modeling pursuit detection in dynamic visual scenes. In L. K. Samuelson, S. L. Frank, M. Toneva, A. Mackey, & E. Hazeltine (Eds.), *Proceedings of the 46th Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Meyerhoff, H. S., Huff, M., & Schwan, S. (2013). Linking perceptual animacy to attention: Evidence from the chasing detection paradigm. *Journal of Experimental Psychology: Human Perception and Performance*, *39*, 1003–1015.

Meyerhoff, H. S., Schwan, S., & Huff, M. (2014a). Perceptual animacy: Visual search for chasing objects among distractors. *Journal of Experimental Psychology: Human Perception and Performance*, *40*, 702–717.

Meyerhoff, H. S., Schwan, S., & Huff, M. (2014b). Interobject spacing explains the attentional bias toward interacting objects. *Psychonomic Bulletin & Review*, *21*, 412–417.

Palmer, S., & Rock, I. (1994). Rethinking perceptual organization: The role of uniform connectedness. *Psychonomic Bulletin & Review*, **1**(1), 29–55.

Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D.,& R Core Team. (2021). nlme: Linear and nonlinear mixed effects models. https://CRAN.R-project.org/package=nlme

Posner, M. I. (1980). Orienting of attention. *The Quarterly Journal of Experimental Psychology*, *32*(1), 3–25.

R Core Team (2023). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. [Computer software]. Retrieved from https://www.R-project.org

Treisman, A. M., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*, *12*(1), 97-136.

Ullman, S. (1984). Visual routines. *Cognition*, *18*, 97–159.

Proceedings of the 23rd International Conference on Cognitive Modelling (ICCM 2025)

Neural Network-Driven Cognitive Models of Phishing Decisions: Evaluating a Privileged Learning Framework for Instance Representation

Elaheh Mehrabi (elaheh@uw.edu)
Department of Industrial & Systems Engineering, University of Washington
3900 E Stevens Way NE Seattle, WA 98195 USA

Prashanth Rajivan (prajivan@uw.edu)
Department of Industrial & Systems Engineering, University of Washington
3900 E Stevens Way NE Seattle, WA 98195 USA

Abstract

Phishing emails target individuals and organizations to compromise security and privacy. Understanding how humans make decisions when encountering these emails is crucial for effective defense. Cognitive models offer a promising approach, but a key challenge lies in representing the contextual information people process, encode, and retrieve—especially in textual form. This study leverages privileged learning to enhance context representation in cognitive models of phishing. Using instance-based learning (IBL) and a neural network-driven text similarity approach, we predict how humans perceive email content and intent. Our results show that this method significantly improves IBL agents' ability to model phishing decisions, advancing personalized anti-phishing defenses.

Keywords: phishing detection; cognitive modeling; privileged learning; instance-based learning; neural networks

Introduction

Phishing attacks deceive individuals into revealing sensitive information or granting unauthorized access to systems. The increasing prevalence of phishing reflects the general increase in the number and severity of cyberattacks (Ulsch, 2014; Li & Liu, 2021). Although early studies attributed phishing susceptibility to a lack of attention to URLs and sender addresses (R. Dhamija, Tygar, & Hearst, 2006), recent research highlights the role of cognitive processes associated with memory retrieval and decision-making (Cranford, Lebiere, Rajivan, Aggarwal, & Gonzalez, 2019; Xu, Singh, & Rajivan, 2022a; Malloy & Gonzalez, 2024).

Familiarity with certain topics or entities could affect how one would assess the legitimacy of an email. For example, when a phishing email contains familiar contexts such as recent conversations or familiar entities, it can evoke a sense of trust, making it difficult for individuals to recognize phishing emails as malicious. Modeling and analyzing how memory processes influence reasoning and decision-making is crucial to understanding why people fall victim to phishing attacks. They are also essential for building next-generation email protection technologies such as personalized anti-phishing training solutions (Azuma, Daily, & Furmanski, 2006) (Newell & Simon, 1972).

Cognitive models grounded in ACT-R and Instance-Based Learning Theory (IBLT) have been used to analyze and predict individuals' responses to phishing emails (Cranford et al., 2019; Xu et al., 2022a). However, the challenge in developing such models lies in *instance engineering*—representation of contextual information that people process, encode, and retrieve when making decisions (Xu & Rajivan, 2024). Previous works have relied on transformer-based language models to construct instance representations, but using embeddings of the entire email text as instances is likely not how people encode and remember emails. (Xu, Singh, & Rajivan, 2022b). Also, representing an entire email as a single embedding within cognitive models could introduce unexplainable errors and biases (Devlin, Chang, Lee, & Toutanova, 2018; Sannigrahi, Genabith, & España-Bonet, 2023).

In this study, we evaluate "Learning Using Privileged Information" (LUPI, (Vapnik & Vashist, 2009)) as an approach to construct input instances for cognitive models of phishing decision-making. This approach enables the use of additional features available during training, which which isn't available during testing (Vapnik & Vashist, 2009; Momeni, Tatwawadi, & Stanford, 2022). In the context of this work, we have 'privileged' data from surveys collected during laboratory experiments containing people's interpretations of key aspects of each email. Such self-reported data would not be available in a real-world environment. So, the idea is to leverage such privileged information to predict how humans would interpret new, unseen emails and use those predicted features as input instances to train and evaluate cognitive models.

Background and Related Work

Susceptibility to Phishing

A significant body of research has studied human susceptibility to phishing attacks. This includes examining whether individuals detect critical cues in phishing emails or websites (e.g., (J. D. Dhamija, Tygar, & Hearst, 2006)), designing interventions to help users focus on suspicious elements in messages (Wu, Miller, & Garfinkel, 2006), and analyzing training programs aimed at teaching phishing detection concepts and strategies (Kumaraguru et al., 2007; Kumaraguru, Sheng, Acquisti, Cranor, & Hong, 2010)). Additionally, many studies have sought to explain why people fall for phishing attempts, drawing on theories of inattention, social influence, and, more recently, memory retrieval

inefficiencies (Cranford et al., 2019; Vishwanath, Harrison, & Ng, 2018; Ferreira & Teles, 2019; Sawyer & Hancock, 2018)). Modeling the memory processes involved in recognizing phishing threats would provide insights into why individuals may be susceptible to specific kinds of attacks.

Modeling Human Response to Phishing

Instance-Based Learning (IBL) models is built on the premise that humans make decisions by drawing from their past experiences (C. Gonzalez, Lerch, & Lebiere, 2003). Prior studies that have used IBL to analyze phishing decisions have relied on using email-specific features (e.g., sender address, subject line, salutation) to represent emails within the models (Cranford et al., 2019; Cranford, Singh, Aggarwal, Lebiere, & Gonzalez, 2021), but these are features relevant to email protocol and delivery and may not fully capture how individuals perceive, interpret and recall emails from memory (Shonman et al., 2022).

Recent research has adopted natural language processing (NLP) techniques with IBL models to model phishing decisions (Xu et al., 2022b). However, it was found that IBL agents may struggle with high-dimensional representations of emails, and relying solely on text features may not be the correct approach to model human decision-making processes. Xu and Rajivan's work (Xu et al., 2022b) highlights the challenges with representing text stimuli as instances, underscoring the need for new approaches that align with how people process email text.

Instance-Based Learning Theory (IBLT): IBLT describes the process of decision-making from experiences (J. F. Gonzalez, Lerch, & Lebiere, 2003). As per IBLT, situations and decisions are conceptualized as instances, which are triads consisting of state, action, and utilities (Xu et al., 2022b). Here, 'state' refers to the characteristics of the environment related to a task/situation (e.g., individual emails), 'action' denotes the decisions made by an individual in each situation (e.g., respond or ignore the email), and 'utilities' represent the anticipated outcomes or benefits that an individual derives from executing such actions.

The IBL agents consist of three primary elements: (1) **Activation:** determining the readiness of memory instances based on recency and frequency (Anderson et al., 2004), (2) **Blending Mechanism**: estimating expected utility based on past experiences (Hakim et al., 2021), and (3) **Similarity:** assessing the resemblance between current and past situations to inform decision-making.

Similarity is a crucial aspect of the IBL model, as suggested by the ACT-R theory (Anderson et al., 2004). Decision-makers recognize a situation according to its similarity to past instances. A similarity rule, defined on situational cues, evaluates the resemblance of previous situations to the current situation, so that the model only considers experiences that occurred in similar situations. In the past, cognitive models of tasks involving decision-making based on text processing typically use large language models such as BERT to represent the text and assess similarities between instances.

BERT for Text Representation: BERT (Devlin et al., 2018) is a transformer-based architecture that uses an attention mechanism to assign weights to words based on their importance in predictions. Its strength in contextual understanding makes it effective for language tasks. In our model, BERT translates email text into high-dimensional vectors for analysis.

Enhancing Cognitive Models with Privileged Learning

Building on prior work (Xu et al., 2022b), we propose integrating privileged learning paradigms into IBL models. Traditional approaches rely solely on textual features, whereas our framework incorporates **survey-annotated data**. Humans do not process emails solely as raw text; they derive meaning based on experience and context. Our approach uses neural networks to predict survey-derived attributes for new emails. Our goal is to develop robust models that accurately predict and respond to emails (including phishing emails), closely reflecting human judgment. By integrating **privileged learning** and **human-annotated data**, we address the challenges identified by Xu and Rajivan (2022) (Xu et al., 2022b).

Method

In this work, we describe a new approach to predict human responses to emails (benign and phishing) by bringing together sentence transformers, neural networks, and Instance-Based Learning (IBL). We specifically evaluate the efficacy of using privileged learning approaches (Vapnik & Vashist, 2009) to generate higher-order, lower-dimensional contextual representations of email text as instances within IBL models (Morrison & Gonzalez, 2024). We compare privileged learning methods with previous approaches that used word embeddings of the entire email text as instance representations. The key idea is to develop neural network-based privileged learning methods to predict features identified by humans as salient to their decision-making. These predicted features are then used as instance representations to train and validate cognitive models of phishing decision-making. Figure 1 shows the overall procedure of building the proposed framework in this study.

Dataset: Two distinct datasets were used to evaluate the performance of our approach. Dataset 1 was used to train and validate the proposed approach. Dataset 2 was used to test the performance of the approach on unseen emails.

The dataset 1 used in this study is derived from a prior laboratory experiment designed to measure the susceptibility of end-users to spear-phishing attacks (Xu et al., 2022b). This experiment utilized a simulated phishing attack environment, comprising groups of four participants. In each group, three participants adopted the roles of end-users, while the fourth participant played the role of an attacker. Participants playing end-user roles were provided with detailed fictitious narratives and were assigned the task of making decisions on a large set of emails on behalf of their respective roles. The emails encompassed various categories, including legitimate,

Survey	Summary
Importance to the persona	Importance
Request for action (Task assigned, click on a link, download attachment, etc)	Action
Request for information or opinion (send a reply message, contact info, send file, image, etc)	Information
Contains status update for an ongoing project or task	Project
Request for a meeting or other communication with you	Meeting
Contains reminder for a meeting, event, or upcoming deadline	Deadline
Spam or marketing or suspicious	Spam
Other	Other

Table 1: Survey questions presented to end-users during each trial.

promotional, mass phishing, and spear-phishing messages. For more details about the experiment design and procedure refer to (Xu, Singh, & Rajivan, 2021)

The dataset 1 comprises a total of 396 emails, each of which was evaluated by a varying number of human raters. The maximum number of raters for a single email was 59, with an average of 18 raters per email. The human users were required to respond to a series of questions about the emails presented to them such as whether they would respond to the email and questions that capture the salient aspects about it. This includes determining whether the email was soliciting a particular action, if it was related to an impending deadline, whether it seemed like a spam message, or if it pertained to a meeting, among other considerations. They are categorized and labeled into seven different classes: "action", "information", "project", "meeting", "spam", "deadline", and "deadline". This dataset was chosen because, to our knowledge, it is the only dataset that includes end-user choices on how they would respond to an email and the email features relevant to human responses. The second dataset consisted of new, unseen emails obtained from another laboratory study(K. Singh, Aggarwal, Rajivan, & Gonzalez, 2020). These were emails that were not used as part of the training phase but were used for testing the performance. This dataset consisted of responses to emails (benign and phishing) from 48 participants. This dataset encompassed a total of 3840 responses to 241 unique emails. On average, each participant responded to approximately 80 emails. Specifically, 55 of the emails were benign, often referred to as "ham" emails, while the remaining 186 emails were identified as "phishing" emails. A key difference between the two datasets is the availability of human-assigned labels for emails in dataset 1.

Feature Extraction and Label Aggregation: We used BERT (Devlin et al., 2018), specifically the 'all-MiniLM-L6-v2' model(Wang et al., 2020) using the SentenceTransformer library to convert the email text into numerical representations, known as embeddings(Mikolov, Karafiát, Burget, Cernocký, & Khudanpur, 2010). These embeddings are lower-order representations and capture the semantic meaning of the emails. The emails in the dataset were preprocessed to remove any null values and passed through the BERT model to generate the embeddings per email. For each email, human-assigned labels and embeddings from BERT were integrated. Since multiple human raters labeled each email, the labels were aggregated for model development and analysis. Aggregation was performed using majority voting. For each email, we created a binary matrix where each row corresponds to an email and each column corresponds to a survey response. The binary matrix is created by comparing the aggregated labels with the total number of responses per email for each survey column. By aggregating the labels per email, we ensure that each email is represented by a single label vector (the majority vote of the labels) and a single feature vector (the mean of the embeddings). Each email has multiple labels that must be predicted, making it a multi-label classification problem.

Model Architecture: We employed sequential neural network model built using the Keras library (Chollet, 2015) to handle the multi-label classification. In multi-label classification, each input can be associated with multiple labels. In this case, each email contains multiple human-assigned labels (from participants' survey responses). The output layer of the neural network used a sigmoid activation function, which means that each output neuron produces a value between 0 and 1, independent of the other neurons. These output values represent the probability of each label being assigned to a given email, allowing the model to predict multiple labels independently. The network consists of three layers:

- An input layer with 64 neurons and a "relu" activation function. The input dimension is equal to the number of features in our dataset (i.e., the length of the embeddings from BERT). The "relu" activation function introduces non-linearity into the model, allowing it to learn complex patterns in the data.
- A hidden layer with 32 neurons and a "relu" activation function. This layer allows the model to learn more complex representations of the data by combining the outputs of the neurons in the previous layer.
- An output layer with a number of neurons equal to the number of survey columns and a "sigmoid" activation function for binary classification. The 'sigmoid' activation

function ensures that the output of each neuron is between 0 and 1, which can be interpreted as the probability of the corresponding class.

We used the Adam optimizer with a learning rate of 0.001. The model was compiled with a binary cross-entropy loss function and a custom accuracy metric, which we refer to as 'fuzzy accuracy'. This metric measures the accuracy within a certain tolerance level (0.3), allowing for some degree of error in the predictions. This is a common approach, particularly when dealing with diverse and subjective data. In our dataset, each unique email has multiple rows of labels assigned by different individuals, resulting in a wide variety of labels for the same email. This reflects the subjective nature of human judgment. By using a 'fuzzy accuracy' metric and allowing for a certain degree of error in the predictions, we aim to capture this variability and provide a more realistic evaluation of our model's performance. Previous studies have highlighted the limitations of accuracy as a sole performance metric in binary classification and how a tolerance level in the accuracy function could affect the performance of binary classifiers(S. Singh & Khim, 2022).

Neural Network Model Training and Evaluation: The model is trained on the training set for 100 epochs with a batch size of 32. After training, we use the model to predict the probabilities for the test set. To prevent overfitting and to ensure that the model generalizes well to unseen data, we incorporate an early stopping mechanism. This mechanism monitors the validation loss during the training process and stops the training when the validation loss has not improved for a specified number of epochs, in this case, 10 epochs. If the validation loss starts to increase, indicating potential overfitting, the training is stopped and the weights from the epoch with the best validation loss are restored. This approach allows us to balance the need for a well-trained model with the risk of overfitting.

Cognitive Model: We leverage the trained neural network to predict labels assigned by humans for each email and use these labels as instance representations of emails within cognitive models. This is in contrast to the previous approach that used embeddings as instance representations.

For each individual in our dataset, we construct a cognitive agent using IBL modeling framework. These agents are trained on a subset of the data corresponding to their respective individuals. The training set consists of emails, corresponding labels, and participant's responses to the emails. The agents' memory is populated with instances from the training set, where each instance is a representation of an email in the dataset and the corresponding human response. During training, when presented with a new email, the agent leverages the previously trained Neural Network to predict a survey vector for the email. This survey vector serves as a lower-dimensional representation of the email, eliminating the need for the agent to interpret the long email text directly. The agent then compares this predicted survey vector with the instances in its memory, based on their similarity. This comparison forms the basis of the agent's decision-making process, allowing it to decide whether to respond or not to an email.

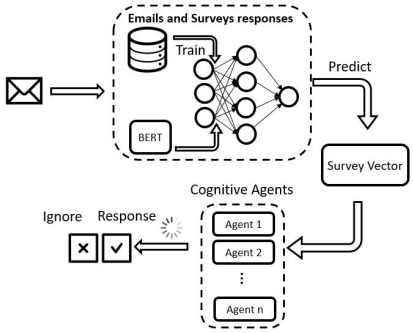


Figure 1: Procedure

Agents Pooled Decision: In our study, we developed a group agent that makes decisions based on the choices of 10 individual agents. Specifically, we utilized 10 individual agents. Each agent was assigned a weight based on their performance, and the group agent made a final decision by considering these weights and the predictions of all individual agents. This decision represents the collective decision on whether to respond or not to an email. The choice of 10 agents was driven by the nature of our dataset. The distribution of emails across raters was uneven, with some raters assigned as few as 2 emails. This was insufficient to train a reliable agent. Therefore, we selected 10 user IDs, each having responded to at least 60 emails, to ensure a robust dataset for each agent's training. This approach helped us to develop agents that could generalize well to new emails.

Baseline Model: To evaluate the effectiveness of our proposed approach, we compare our results against a baseline model that utilizes a standard BERT-based representation without incorporating user perception features. This baseline model was developed in prior work on cognitive modeling by Xu and colleagues (Xu et al., 2022b). The primary objective of this model was to represent email text in a way that could be used within cognitive agents to predict human responses to phishing and benign emails.

In this baseline approach, email content was encoded using a pre-trained BERT model that generates fixed-length embeddings capturing the semantic meaning of the text. These embeddings were then used directly as input representations for the IBL cognitive agents, which made response decisions based on the similarity between the current email and past instances stored in memory. Unlike our proposed method, which integrates privileged learning and higher-order contextual representations derived from human perception, the baseline model relied solely on textual embeddings from Sentence-BERT (nli-distilroberta-base-v2)(Reimers & Gurevych, 2019).

This baseline achieved, on average, less than 60% accuracy in predicting human participant responses to phishing emails.

While BERT has demonstrated strong performance in various NLP tasks, using raw embeddings alone for phishing decision modeling did not fully capture the contextual factors influencing human decision-making.

Results

Neural Network Performance: Following the privileged learning approach, the neural network was designed to predict survey vectors for emails, and its performance was evaluated over several epochs. The training and validation loss, as well as the accuracy, were recorded for each epoch.

Although we initially defined 100 epochs for the training process, we incorporated an early stopping mechanism to prevent potential overfitting. This mechanism monitors the validation loss during the training process and stops the training when the validation loss has not improved for a specified number of epochs, in this case, 10 epochs. As a result of this early stopping mechanism, the training process was halted at the 40th epoch.

Over the 40 epochs, the model demonstrated a consistent decrease in validation loss, indicating that the model was learning effectively from the training data (See Figure 2). The accuracy also showed improvement over time. By the final epoch, the model achieved a training accuracy of 86.97 and a validation accuracy of 87.09. From the figure, we can see that the neural network was able to accurately predict survey vectors for the emails in both the training and validation sets. The close alignment of the training and validation accuracy also suggests that the model generalizes well to unseen data, which is a crucial property for practical applications.

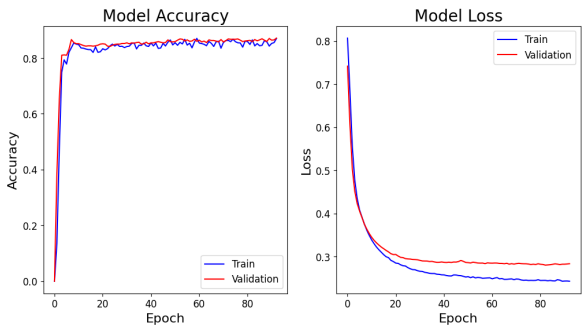


Figure 2: Model Performance Metrics on Two Datasets

Overall, the performance of the neural network in predicting survey vectors for emails was satisfactory. The consistent decrease in validation loss and the high accuracy achieved by the final epoch demonstrate the effectiveness of the neural network in transforming high-dimensional email data into survey vectors. This transformation is a critical step in our approach, enabling the cognitive models to operate in a more manageable, lower-dimensional space. The early stopping mechanism ensured that the model is not only accurate but also robust and capable of generalizing to new, unseen data.

Cognitive Agent Performance: We evaluated the performance of cognitive agents in predicting human responses to emails using standard metrics such as accuracy, precision, recall, and F1 score. The performance of these cognitive agents was assessed using two distinct datasets. The first dataset, which is used for training the agents, was divided using an 80-20 train-test ratio. This division implies that 80% of the data is utilized for training the agents, while the remaining 20% is reserved for testing their performance. This allows us to evaluate how well the agents have learned to classify emails based on the training data. The second dataset was used to test the ability of the agents to generalize their learning to new, unseen emails.

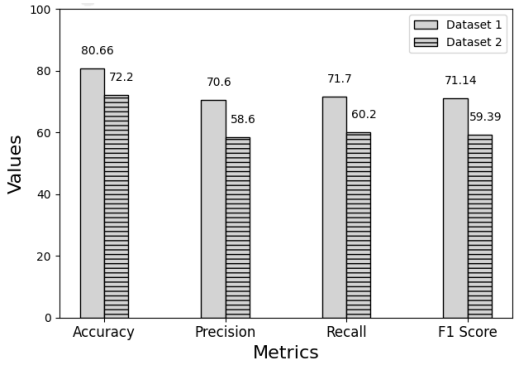


Figure 3: Model Performance Metrics on Two Datasets

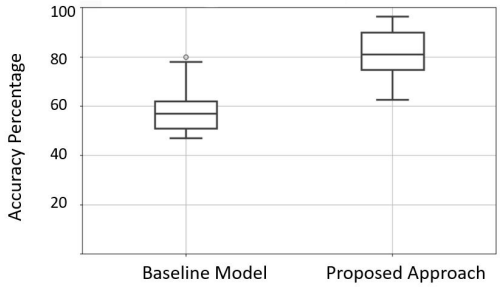


Figure 4: Accuracy for the Group of Trained IBL Agents over two Approaches

As shown in Figure 3, on the first dataset, the agents achieved an accuracy of 80.66, a precision of 70.6, a recall of 71.7, and an F1 score of 71.14. On the second dataset, the agents achieved an accuracy of 72.2, a precision of 58.6, a recall of 60.2, and an F1 score of 59.39. The model's performance is encouraging, demonstrating significant improvement over the baseline model. The baseline model, which utilized the full text of the emails within the cognitive agents, achieved an accuracy of approximately 60 percent in its predictions. Figure 4 presents two box plots showing the accuracy distribution across individual IBL agents, further illustrating that the proposed model consistently outperforms the baseline, achieving higher accuracies across single agents in

the group. These metrics provide a detailed view of the performance of our cognitive agents, both in terms of their ability to classify emails based on the training data and their ability to generalize their learning to new, unseen instances.

In summary, our results demonstrate the effectiveness of our approach in predicting human responses to emails in laboratory experiments. We have found that sequential neural network models are able to predict human-assigned labels from word embeddings of email text derived from language models such as BERT. We have also found that using such predicted human-assigned labels as representations of context in instances enable cognitive models to better predict human responses to emails, not only on the dataset it was it was trained upon but also generalizes to good extext when applied to new, unseen data. This suggest the approache's ability to generalize.

Discussion

We found that representing decision-making context using lower-dimensional vectors with factors salient to human response would be effective. This work introduces privileged learning as an approach for generating such low-dimensional, salient factors from email text embeddings. This differs from earlier methods for constructing instances that relied either on manually coded context features or using numerical embeddings of the full email text as instance representations. The former approach of using fixed, hand-coded features would not be scalable or generalizable given that over 400 billion emails are exchanged daily. Although the latter approach is scalable, it is not representative of how humans encode email instances into memory, resulting in cognitive agents that are ineffective at accurately predicting human responses to emails (Xu et al., 2022b).

Our work demonstrates that neural network models can be trained to predict email features labeled by humans to be salient to their responses (a multi-class prediction problem) using text embeddings generated by language models. We also found that using these predicted factors as input instances is significantly better than using raw word embeddings as input for cognitive models of phishing decision-making. Although the finding that using contextual features salient to humans as inputs results in more effective cognitive models is intuitive, our work shows that we could develop models that can effectively predict such features on unseen emails. Such methods enable future work to automate the process of generating low-dimensional contextual salient features for developing cognitive models. This work could enable future work to determine salient features from a small sample of human subjects and emails and use them to develop models that can generalize to unseen, larger datasets. This approach essentially aims to overcome the challenges of designing instance representations for cognitive models of phishing decision-making and facilitates the development of more scalable and generalizable models.

When applied to the second dataset with unseen emails, the model's performance, while not as effective as on the first dataset, was still adequate (see Figure 3). Although both datasets contained human responses to emails (benign and phishing), the second dataset is distinct in terms of email topics and distribution. Importantly, the features collected through the survey from participants, as included in dataset 1, are not exhaustive. The first dataset contains only labels indicating a limited subset of email intentions, such as whether an email includes "information request," "meeting request," and "reminders for event," which are considered relevant to human responses to emails. However, it does not account for other potentially important features relevant to humans, which they are likely to encode in memory, such as the sender's name, the type of action required, and other intentions not captured in the dataset. Future research could focus on developing a framework of such features salient to human response to emails which could be used to develop more robust and generalizable cognitive models of phishing decision-making.

Our findings suggest that defenses against phishing threats could be significantly enhanced by adopting cognitive models to measure individuals' susceptibility to phishing and to develop personalized training programs. This work highlights the need for advancements in methodologies capable of constructing scalable instance representations of emails to train and refine cognitive models for phishing decision analysis.

We demonstrate privilege learning as an effective approach to generate low-dimensional, human-relevant features from email text, providing a practical method for constructing input instances for cognitive models. We hope this work will inspire further efforts in this area. By focusing on the cognitive aspects of user behavior, we gain deeper insights into how individuals perceive and respond to potential phishing emails. This cognitive-centered perspective is crucial for understanding user decision-making processes and for designing more effective, tailored anti-phishing training strategies.

One of the limitations of our study lies in the labeling scheme used in dataset 1. The predefined labels, while useful, may not fully capture the nuances of human decision-making processes. These labels are derived from survey questions responded to by human users during the experiment. However, these questions may not be universally applicable across different datasets or fully represent all aspects of emails that humans pay attention to when making decisions. This could potentially limit the effectiveness of our feature vectors.

Another limitation lies in the use of lab-based datasets for testing our model. While these datasets are valuable for initial testing and development, they are limited in the diversity and complexity observed in real-world email based phishing attacks. Consequently, the robustness and generalizability of our model could be further tested on more diverse and larger datasets in future work. This would provide a more rigorous test of cognitive model's ability to adapt to different environments and model responses to a wider range of phishing attacks.

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.

Azuma, R., Daily, M., & Furmanski, C. (2006). A review of time critical decision making models and human cognitive processes. In *2006 ieee aerospace conference* (pp. 9–pp).

Chollet, F. (2015). *Keras: Deep learning library for python.* (GitHub Repository, Retrieved from `https://github.com/fchollet/keras`)

Cranford, E. A., Lebiere, C., Rajivan, P., Aggarwal, P., & Gonzalez, C. (2019). Modeling cognitive dynamics in end-user response to phishing emails. In *Proceedings of the 17th iccm.*

Cranford, E. A., Singh, K., Aggarwal, P., Lebiere, C., & Gonzalez, C. (2021). Modeling phishing susceptibility as decisions from experience. In *Proceedings of the 17th international conference on cognitive modeling (iccm).*

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*.

Dhamija, J. D., Tygar, J. D., & Hearst, M. (2006). Why phishing works. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 581–590).

Dhamija, R., Tygar, J. D., & Hearst, M. (2006). Why phishing works. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 581–590).

Ferreira, A., & Teles, S. (2019). Persuasion: How phishing emails can influence users and bypass security measures. *International Journal of Human-Computer Studies*, *125*, 19–31.

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

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

Hakim, Z. M., Ebner, N. C., Oliveira, D. S., Getz, S. J., Levin, B. E., Lin, T., & Wilson, R. C. (2021). The phishing email suspicion test (pest): A lab-based task for evaluating the cognitive mechanisms of phishing detection. *Behavior Research Methods*, *53*, 1342–1352.

Kumaraguru, P., Rhee, Y., Acquisti, A., Cranor, L. F., Hong, J., & Nunge, E. (2007). Protecting people from phishing: The design and evaluation of an embedded training email system. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 905–914).

Kumaraguru, P., Sheng, S., Acquisti, A., Cranor, L. F., & Hong, J. (2010). Teaching johnny not to fall for phish. *ACM Transactions on Internet Technology (TOIT)*, *10*(2), 1–31.

Li, Y., & Liu, Q. (2021). A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments. *Energy Reports*, *7*, 8176–8186.

Malloy, T., & Gonzalez, C. (2024). Applying generative artificial intelligence to cognitive models of decision making. *Frontiers in Psychology*, *15*, 1387948.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech* (Vol. 2, pp. 1045–1048).

Momeni, A., Tatwawadi, K., & Stanford, U. (2022). *Title of the paper is not provided.* (Journal/Conference name and page numbers are not provided)

Morrison, D., & Gonzalez, C. (2024). *Pyibl: Python implementation of ibl.* (Journal/Conference/Book title and page numbers not provided)

Newell, A., & Simon, H. (1972). *Human problem solving*. Oxford, England: Prentice-Hall.

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint*.

Sannigrahi, S., Genabith, J. V., & España-Bonet, C. (2023). Are the best multilingual document embeddings simply based on sentence embeddings? *arXiv preprint*.

Sawyer, B. D., & Hancock, P. A. (2018). Hacking the human: The prevalence paradox in cybersecurity. *Human Factors*, *60*(5), 597–609.

Shonman, M., Shi, X., Kang, M., Wang, Z., Li, X., & Dahbura, A. (2022). Using a computational cognitive model to understand phishing classification decisions. In *35th international bcs human-computer interaction conference* (pp. 1–10).

Singh, K., Aggarwal, P., Rajivan, P., & Gonzalez, C. (2020, December). What makes phishing emails hard for humans to detect? In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 64, pp. 431–435). Los Angeles, CA: SAGE Publications.

Singh, S., & Khim, J. T. (2022). Optimal binary classification beyond accuracy. In *Advances in neural information processing systems 35* (pp. 18226–18240).

Ulsch, M. (2014). *Cyber threat!: How to manage the growing risk of cyber attacks*. John Wiley & Sons.

Vapnik, V., & Vashist, A. (2009). A new learning paradigm: Learning using privileged information. *Neural Networks*, *22*(5), 544–557.

Vishwanath, A., Harrison, B., & Ng, Y. J. (2018). Suspicion, cognition, and automaticity model of phishing susceptibility. *Communication Research*, *45*(8), 1146–1166.

Wang, W., Bi, B., Yan, M., Wu, C., Xia, J., Li, Z., ... Si, L. (2020). Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint*.

Wu, M., Miller, R. C., & Garfinkel, S. L. (2006). Do security toolbars actually prevent phishing attacks? In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 601–610).

Xu, T., & Rajivan, P. (2024). *Analyzing instance representation in cognitive models of phishing decision making.*

Xu, T., Singh, K., & Rajivan, P. (2021). Spearsim: Design and evaluation of synthetic task environment for studies on

spear phishing attacks. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 65, pp. 1500–1504). SAGE Publications Sage CA: Los Angeles, CA.

Xu, T., Singh, K., & Rajivan, P. (2022a). Modeling phishing decision using instance based learning and natural language processing. In *Proceedings of the hawaii international conference on system sciences (hicss)* (pp. 1–10).

Xu, T., Singh, K., & Rajivan, P. (2022b). Modeling phishing decision using instance based learning and natural language processing. In *Proceedings of the hawaii international conference on system sciences (hicss)* (pp. 1–10).

Model of Anxiety Behavior Based on Time Perception and Anticipation in ACT-R

Kazuma Nagashima[1,2] (nagashima.kazuma.16@shizuoka.ac.jp)
Junya Morita[2] (j-morita@inf.shizuoka.ac.jp)

[1]Japan Society for the Promotion of Science (JSPS)
[2]Department of Informatics, Shizuoka University, 3-5-1 Johoku, Chuo-ku, Hamamatsu, 432-8011 Japan

Abstract

Humans live in the flow of time. However, the perception of time varies between individuals and changes depending on the situation. This variability is believed to be closely related to emotions. For example, people tend to perceive time as passing quickly when they are having fun, whereas they may feel it slows down when they are anxious. Based on this relationship, this study reports a simulation examining the effects of anxiety on perceptual-motor tasks from the perspective of time perception. Previous research on time perception suggests that anxiety increases arousal, which in turn makes time feel like it is passing more slowly. In particular, individuals with high trait anxiety are more sensitive to environmental changes, leading to more pronounced distortions in time perception. Considering these findings, this study conducted a simulation using the cognitive architecture ACT-R to investigate how anxiety affects perceptual-motor tasks. The modeling of anxiety in this study is based on the anticipation of failure-related memories. The simulation results confirmed that freezing behavior occurred in response to the task, affecting the model's performance.

Keywords: time perception; anxiety; cognitive modeling; ACT-R

Introduction

Humans live in the flow of time. While objective time is defined in the world, our subjective time is far from uniform (Zakay, Block, et al., 1995). Perceptions of the time flow vary from individual to individual and depend on the situation and psychological state. The relationship between emotion and time perception is widely recognized in psychological and neuroscientific research. For instance, time seems to pass quickly when we have an enjoyable time, whereas it appears to slow when we are anxious or stressed (Stetson, Fiesta, & Eagleman, 2007). These phenomena indicate that the perception of time is closely related to our emotional and attentional states.

To further explore the mechanisms underlying time perception, some researchers have utilized cognitive architectures, which are general frameworks designed to replicate human cognitive processes with memory and physiological parameters. Among these frameworks, the most versatile and widely used is ACT-R (Adaptive Control of Thought-Rational: Anderson, 2007), as shown by Kotseruba and Tsotsos (2020). ACT-R includes modules for memory and perception, as well as a module for handling time perception (Taatgen, Van Rijn, & Anderson, 2007).

Although ACT-R has a temporal module for modeling cognitive processes, it lacks a module dedicated to handling emotions. Given the aforementioned empirical link (Stetson, 2007), the influence of emotions on time perception should be incorporated into ACT-R. We believe that such research connecting the cognitive module and emotions will contribute to the recent discussion of an emotions module in cognitive architectures (Rosenbloom et al., 2024).

In particular, it can be assumed that the relationship between emotions and time perception may involve bidirectional interactions rather than a simple one-way influence. For example, not only can stress and anxiety alter time perception, but the way time is perceived can also affect stress levels and the allocation of attention. To clarify such interactions, it is essential to develop integrated models of emotions and time perception.

To address the aforementioned issues, this study aims to incorporate the influence of emotion into a model of time perception using the ACT-R. This approach not only deepens our understanding of time perception but also allows for a more detailed elucidation of how anxiety and stress impact decision-making and learning. In the following sections, before introducing our proposed model, we first review relevant research on time perception and cognitive models that have explored this topic.

Related Works

As background to this study, we first review basic theories and experimental findings related to human time perception. Then, we describe the research on which the model in this study is based.

Human Time Perception and Emotions

As components that constitute human time perception, Rakitin et al. (1998) pointed out the involvement of an internal clock, working memory, and decision-making processes. Furthermore, Matell and Meck (2000) extended this model from a neuropsychological perspective and examined the brain regions involved in time perception. As a result, it was revealed that short time intervals (on the order of milliseconds) and long time intervals (ranging from seconds to minutes) are controlled by different mechanisms. Short time intervals are processed unconsciously, whereas long time intervals are closely related to working memory and attention.

In the mechanism of time perception, the allocation of attention is considered to play a particularly important role.

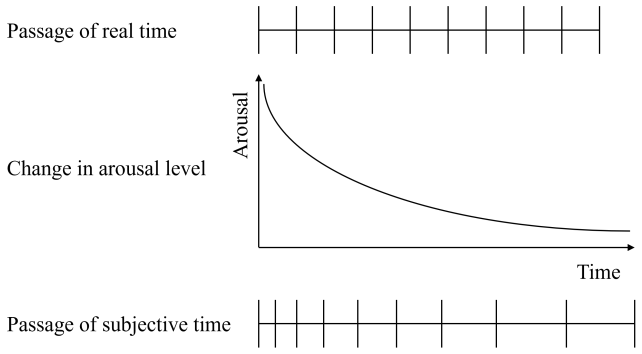


Figure 1: Relationship between arousal level and subjective time.

Zakay et al. (1995) demonstrated that the way attention is directed affects the accumulation of pulses and proposed this theory as the Attentional Gate Model (AGM). According to this model, when attention is directed towards the passage of time, the gate of the internal clock opens, and more pulses are counted, leading to a longer subjective estimation. On the other hand, when attention is directed towards other events or loose concentration, the gate tends to close, reducing the number of pulses counted. Consequently, a shorter subjective estimation is reached. Thus, arousal levels strongly influence the allocation of attention, which in turn affects time perception. For example, as shown in Figure 1, when arousal is low, the accumulation of pulses slows down, making time feel like it passes quickly. Conversely, when arousal increases, the accumulation of pulses of the internal clock rises, making the passage of time feel slower.

Arousal is closely related to emotions, and consequently, time perception is also affected (Droit-Volet & Meck, 2007). For example, when one is engrossed in enjoyable activities, arousal moderately increases, making time feel shorter (Csikszentmihalyi, 1990). This is because, during enjoyable activities, attention is directed towards external events, weakening the process of counting time, resulting in a subjective shortening of time. On the other hand, when arousal is excessively heightened due to stress or anxiety, the process of counting time is strengthened, making the passage of time feel slower (Fayolle, Gil, & Droit-Volet, 2015). This effect is more pronounced in individuals with high trait anxiety (Spielberger, Gonzalez-Reigosa, Martinez-Urrutia, Natalicio, & Natalicio, 1971). as they tend to have a stronger anxiety disposition (Bar-Haim, Kerem, Lamy, & Zakay, 2010).

Such changes in time perception during anxiety are considered to be an evolutionarily adaptive mechanism (LeDoux & Pine, 2016). For example, rapid behavioral choices such as freezing or flight are performed when detecting danger. In general, such a choice of behavior in a moment can be considered to be generated from past memories, but external stimuli do not merely trigger this process; it is amplified by internal anticipation.

Cognitive Models Related to Time Perception and Emotions

The mechanism of time perception has also been a subject of modeling within the cognitive architecture ACT-R. The temporal module proposed by Taatgen et al. (2007) follows the AGM framework and simulates the delay in time perception caused by attentional shifts. This module has been incorporated into the standard ACT-R modules and has been utilized in models related to gaming tasks requiring rapid key operations (Anderson, Betts, Bothell, Hope, & Lebiere, 2019; Gianferrara, Betts, & Anderson, 2021), as well as driving and multitasking (Kujala & Salvucci, 2015; Nagashima, Nishikawa, & Morita, 2024).

Furthermore, Nagashima, Nishikawa, Yoneda, Morita, and Terada (2022) proposed a model that integrates the module with mind-wandering induced by fluctuations in arousal. The model further incorporates motor learning (Anderson et al., 2019) and physiological mechanisms (Dancy, Ritter, Berry, & Klein, 2015) to explain how states of distraction emerge. However, their model only accounts for mind-wandering due to fluctuations in arousal and does not reproduce behavioral changes under anxiety. To incorporate the effects of anxiety into an ACT-R model, it is necessary to introduce a process of leading anxiety, not only representing changes in arousal or emotional changes. Thus, a mechanism of anticipating negative future events needs to be represented in a model.

Model

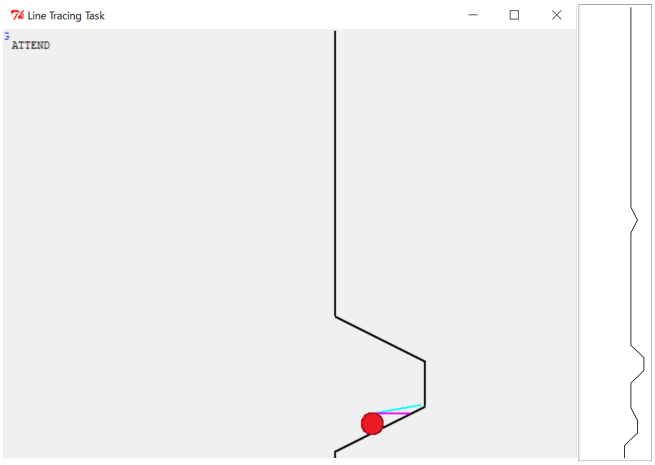
The model in this study represents the process of anxiety by adding the mechanism of anticipating failure (recalling failure memories) to the model proposed by Nagashima et al. (2022). In the following sections, we present a task used by Nagashima et al. (2022), followed by a description of the model.

Task

The line-following task examined by Maehigashi, Miwa, Terai, Kojima, and Morita (2013) was used in this study. Several ACT-R models (Morita et al., 2020; Nagashima et al., 2022) have been constructed for this task in the past. In particular, Nagashima et al. (2022)'s model integrates the basic perceptual-motor cycle with the temporal module (Taatgen et al., 2007), which aligns with the purpose of this study. By using this model, this study aims to examine how emotions in time affect perceptual-motor processes.

Figure 2 shows the task environment in this study. The left side represents the screen that the the model perceives, while the right side shows the overall course that the model observes during each round of the task. In the screen, the circular object (Vehicle) and the black line (Line) are presented. Also, the cyan line drawn from the top of the vehicle indicates the distance to the goal (refraction points of the line), and the magenta line shows the distance to the black line.

By presenting such a screen, the task requires the model to operate the vehicle to follow the line that scrolls from the top



Screen shot of the window

Figure 2: Task interface. Screenshot of task window (left) and overall view of the line (right).

to the bottom of the screen. The line changes along the course (1500 pixels) on the right side of Figure 2. These courses involve connecting patterns of lines with a height of 48 pixels at angles of 30, 45, 90, 135, and 150 degrees. To enhance the model's motor learning, the task is designed with mostly consecutive 90-degree patterns.

The screen updates at 25 fps, with the course moving one pixel downward with each update. Thus, the course takes one minute to complete scrolling, and when the model reaches the end of the course, the same course begins to scroll again. Hereafter, this one-minute repetition is defined as a round.

The vehicle moves when the specified key is pressed. Each key press during screen refresh shifts the vehicle two pixels in the indicated direction, preserving its alignment with the scrolling 30 and 135-degree lines. When following these lines, holding the key allows smooth tracking. However, on a 45 or 135-degree course, intermittent key presses are needed to avoid excessive movement. Since the vertical scrolling is automatic, the model focuses on moving the vehicle left and right to stay on the line.

The task duration was 30 minutes, with the same course repeated 30 times. Additionally, during the task, the probe asking the state of concentration on the task was presented at approximately 50-second intervals.

Module and Process

The aforementioned task can be modeled as a simple repetition of perceptual and motor processing, as shown in Figure 3. This cycle is based on the model constructed in previous studies (Morita et al., 2020; Nagashima et al., 2022). The most significant update in this study is the addition of a path from Objective to Perception, which is triggered by the recall of anxiety memories. These processes are realized by the functions implemented in the following modules.

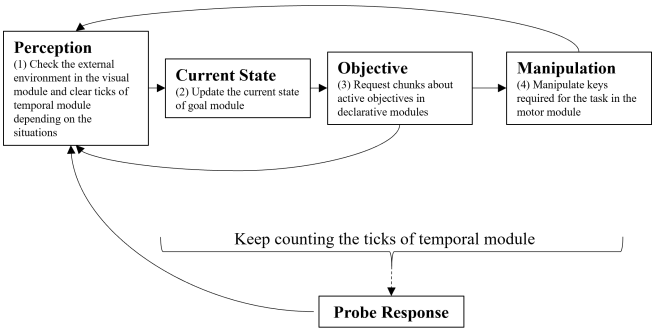


Figure 3: Block diagram showing the model processing.

Visual Module This module is utilized for interaction with the external environment. Specifically in this study, this module reads information necessary for performing the task (e.g., the position of vehicles or the angles of lines) from the screen (Figure 2). For example, Figure 2 shows the distance to the goal and the line as cyan and magenta lines. By reading the length of these lines, the model recognize a task situation for decision-making.

Motor Module This module simulates physical operations necessary for tasks. The model in the current study uses it to performs key presses corresponding to the movement of the vehicle and responses to probes.

Goal Module This module stores states to control flows in a task. In this study, the module maintains the left-right relationship between the vehicle and the scrolling line as read by the visual module. Additionally, the module stores flags for success (online) and failure (offline) of the task based on the positional relationship between the line and the vehicle. This flag is updated in each cycle.

Declarative Module This module stores chunks, which are units of symbolic information in ACT-R. In general, ACT-R chunks represent episodic memory, semantic knowledge, and goal-related memory. Among these, goal-related memory is essential for expressing anxiety in this model. It is not only provided by the task but also constructed by memorizing past states of the goal module. Specifically, the flag values of goal-related chunks in each cycle (success: online / failure: offline) are memorized. The model retrieves these memories each cycle to verify its current goal. The flag values serve as anticipations for the task's future state (success or failure).

Production Module This module manipulates the other modules by selecting and applying production rules using states held by the other modules. In the current model, the application of this module results in the flow shown in Figure 3. Importantly, each transition (corresponding to firing a production rule) takes a specific time cost (50 ms), following the default setting of ACT-R. By accumulating these time costs,

the model can predict the line-following performance in human compatible time constraints. Specifically, the modules shown so far are integrated in the following steps:

1. The model sees the external environment in the visual module (Figure 3 (1))
2. The current state of the goal module is updated (Figure 3 (2))
3. A goal chunk is requested to the declarative module (Figure 3 (3))
4. An operation (key press) for the task is performed through the motor module (Figure 3 (4))

After the above steps, the visual module checks for a new state in the external environment and returns to step 1. In step 3, the model retrieves chunks related to previous goals. This recall is involved in the transition to step 3 of the next cycle. If the model retrieves a memory of a successful task, it transitions to step 4 and performs the operations. If the model retrieves a memory of a failed task, it transitions to step 1. We further explain the mechanism for switching between these two loops in the next section.

Anxiety Processes

In this study, the process of anxiety is expressed as freezing behavior (LeDoux & Pine, 2016). This corresponds to the direct path from step 3 back to step 1 in Figure 3. When anxiety increases, the frequency of selecting this path is supposed to increase, causing the model to repeatedly check the situation without performing any task-related operations. This process is modeled through the interaction between time perception under anxiety and the anticipation (recall) of failure. This section provides a detailed explanation of these mechanisms.

Changes in Time Perception and Arousal Levels As previously mentioned, changes in arousal levels affect time perception. This study adopts Nagashima et al. (2022)'s model to simulate this arousal dynamics. The idea is originated from ACT-Rϕ (Dancy et al., 2015), which interprets ACT-R parameters from the perspective of physiological mechanisms, such as homeostasis. The fundamental idea is to position physiological mechanisms as modulators that adjust the cognitive processes of ACT-R. Based on this idea, various physiological indicators are assumed to correspond to numerical parameters in ACT-R. A representative example of this correspondence is the relationship between the amount of epinephrine released during sympathetic nervous system activation (arousal state) and ANS (activation noise s), one of ACT-R's noise parameters (Dancy et al., 2015).

The ANS parameter is used in the activation of ACT-R calculated by the following formula:

$$A_i = B_i + \varepsilon_i \tag{1}$$

where A_i is the activation value of chunk i, B_i is the base level, and ε_i is the noise generated from the ANS parameter. The base level, which represents frequency and recency, is calculated as:

$$B_i = \ln\left(\frac{n}{1-d}\right) - d \cdot \ln(L) + \beta_i \tag{2}$$

where n is the reference-count (the number of times the chunk has been recalled), L is the creation time (the elapsed time since the chunk was created), d is the decay parameter, and β_i is the baseline value added to the base level.

As this equation shows, the base level directs the recalled memory conversion (exploit). In contrast, the ANS works toward diversifying the model's behavior (explore). This behavior allows us to understand the model's arousal state in relation to the ANS. In the model by Nagashima et al. (2022), the ANS is adjusted based on this concept (small ANS with high arousal, large ANS with low arousal), representing the fluctuations in arousal levels as the task progresses.

To simulate arousal dynamics, Nagashima et al. (2022) used ACT-R's temporal module (Taatgen et al., 2007). As noted earlier, time perception is influenced by attention directed toward the task. As shown in Figure 1, time feels slower under high arousal and faster under low arousal. This suggests that arousal dynamics can be represented through the temporal module, which is a part of a more integrated architecture.

Time perception in ACT-R is controlled by a mental timer (pacemaker). This timer counts the number of ticks (t) that have elapsed since it started, using the equation

$$t_n = a \cdot t_{n-1} + \varepsilon_2, \tag{3}$$

where a stochastic noise (ε_2) is added for each count (n). This relationship explains why estimates for long time intervals are less accurate than estimates for short time intervals.

In addition, we assumed that the decrease in the accuracy of time perception corresponded to a decrease in arousal over time. Specifically, we introduced the equation

$$\varepsilon_i = k \times t \tag{4}$$

where k is a coefficient to adjust the decrease in arousal level with respect to time. In other words, the noise in memory recall increases from the point the timer is set until it is reset. In this study, we map the diversification of behavior induced by this increase to a decrease in arousal level. Conversely, the increase in arousal level corresponds to the timer reset. With continuous refresh of the timer, the model "correctly" sets a goal with a higher activation that should be executed in the task.

Anticipation of Failure As mentioned earlier, this study defines anxiety-related behavior as freezing (LeDoux & Pine, 2016). Freezing occurs when a memory representing a failed goal is recalled. Therefore, as such memories accumulate, task failures increase. We also assume that an individual's anxiety trait can be represented as a bias in stored memories

at the start of the task (Van Vugt, Van Der Velde, & Investigators, 2018). Specifically, it corresponds to the difference in activations between success and failure memory chunks. The recall of these memories influences the model's behavior. When recalling a success memory, the model performs task-related operations (Manipulation), whereas recalling a failure memory causes it to return to the Perception state without executing any operations.

The above memory recall process can be assumed to be influenced by arousal levels. The model represents arousal levels using the timer (equation 4). As pointed out in previous studies (Bar-Haim et al., 2010; Fayolle et al., 2015), it has been noted that human time perception slows down in anxiety-inducing situations. This study represents this phenomenon through the temporal module integrated into ACT-R. In the timer of the temporal module, when the number of ticks is small, the intervals between counts become shorter, and the rhythm of the internal clock speeds up, making time feel slower. Conversely, when the number of ticks is large, the intervals between counts become longer, and the internal clock slows down, making time feel faster.

This study applies this mechanism to the model of anxiety. We assumed that in a normal condition (not in a pathological condition), the activation value of successful memories is higher than that of failed memories (people are positively biased). When arousal levels are low (a large number of ticks), memory noise in ACT-R increases, making the recall of failed memories more likely. Conversely, when arousal levels are high (a small number of ticks), noise decreases, reducing the likelihood of recalling failed memories. Incorporating the effects of arousal on memory recall in this way allows for a more appropriate modeling of anxiety's influence on task performance.

Simulation

We examined how parameters related to time perception and anxiety affect the behavior of the constructed model.

Setting

As a parameter related to time perception, the value of k in equation 4 was set to two levels (0.03 and 0.06). When this value is small, the model's time perception becomes accurate, leading to a high arousal state. Conversely, when this value is large, time perception becomes inaccurate, leading to a low concentration state.

Additionally, the timer in ACT-R, as shown in equation 3, is reset at the following timing:

- At the start of offline
- During offline
- At the start of online
- At the probe response

In other words, we assumed that the model's concentration (arousal level) recovers when it detects a task failure or when there is an external stimulus (probe input).

Furthermore, this study manipulates anxiety traits based on memory bias. The ease of recalling failed memories is manipulated by the difference in activations between success and failure chunks related to goals before the task begins. In this study, the reference-count for success chunks is fixed at 800, while the reference-count for failure chunks is manipulated to 400, 600, and 800. As mentioned earlier, the difference in activations calculated from these counts is assumed to reflect the model's anxiety traits. A larger difference corresponds to more optimistic thinking, while a smaller difference corresponds to more pessimistic thinking. Other parameter settings follow the model from previous research (Nagashima et al., 2022).

The model's behavior was examined by the following indicators:

(a) ANS: Value calculated by equation 4. This corresponds to the state of concentration.

(b) Freezing ratio: Percentage of Freezing Behavior. This is behavior associated with anxiety.

(c) Offline ratio: Percentage of time where the vehicle does not follow the line. Low values indicate poor performance. This is a measure of overall task performance.

Result

We ran 100 simulation runs and summarized the results in Figure 4. As expected by equation 4, the ANS was higher when the k was high (0.06) compared to when it was low (0.03). On the other hand, there was no large difference caused by the reference-count of failure memories stored in the declarative module beforehand (though there are small differences at $k = 0.03$). In other words, whether the recalled memory was of success or failure had little effect on the increase in ticks, confirming that the degree of noise primarily depends on the value of k.

A similar effect of arousal due to time perception is also evident in the Freezing ratio, which represents the impact of anxiety on behavior. When memory recall variability was high and arousal was low ($k = 0.06$), the Freezing ratio was higher compared to when memory recall variability was low and arousal was high ($k = 0.03$). Additionally, we can find an increase of Freezing under the condition where the activation value of failure memory was high (reference-count = 800) at $k = 0.03$, although the effect was not prominent at $k = 0.06$. This suggests that under high-arousal conditions, failure memories continued to be reinforced, leading to an increase in anxiety-related behavior.

In the performance measure of offline ratio, a similar effect of k values was observed, as seen in previous indicators. Specifically, the greater the variability in memory recall, the less likely the model was to recall "the correct task goal," leading to a higher offline rate. However, this effect

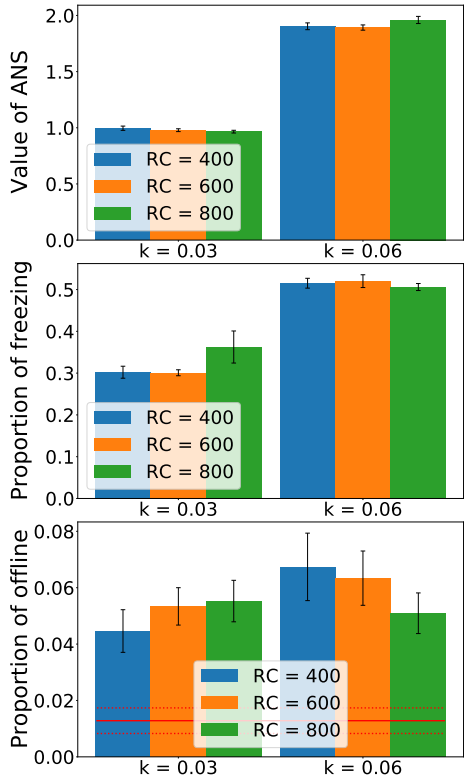


Figure 4: Results of the 100 simulation runs for each condition. The x-axis represents the value of k in equation 4, and the y-axis shows the values of each indicator. Each bar represents the reference-count (RC) of failure memories for the task accumulated in declarative knowledge beforehand. The error bars indicate the Standard Error (SE). Additionally, the red solid line for the Offline ratio represents human averages of a similar simple course task (Nagashima et al., 2024) ($n = 24$). The upper and lower dashed lines indicate the SE.

varied depending on the strength of anxiety-related memories. When arousal was high ($k = 0.03$), the most optimistic model (reference-count = 400) performed better on the task, whereas its performance deteriorated under low-arousal conditions, resulting in a higher offline rate. Conversely, models that were more likely to recall failure memories exhibited poorer task performance under high-arousal conditions but performed better when arousal was low. This suggests that the heightened activation of failure memories under high arousal led to increased Freezing. Although the performance decreases due to memory recall variability under low-arousal conditions remains somewhat puzzling. One possible explanation is that the dispersion of activation prevented an excessive focus on failure memories.

Conclusion

This study constructed a model integrating time perception and emotion in the cognitive architecture ACT-R. Among various emotions, this study focused on anxiety, setting up the processes of anticipating failure (recalling failure experiences) and freezing behavior (LeDoux & Pine, 2016). It incorporated the effect of this recall on time perception. This model, while enhancing arousal connected to time perception, induced anxiety-related freezing behavior through the manipulation of recall noise and the strength of failure memories. This behavior simultaneously showed both positive and negative effects on task achievement. Therefore, it suggests that balancing anxiety and arousal is crucial for optimal performance. These results are consistent with the theory of optimal arousal levels (Yerkes & Dodson, 1908).

We consider that the significance of this study lies in integrating modules in cognitive architecture to represent anxiety as an emotion. Our model is novel in its approach to expressing emotional behavior through time perception. The findings contribute not only to advancing time perception research in cognitive science and psychology but also to extending cognitive architectures that account for emotional influences. Future work could expand this model to incorporate time perception changes related to emotions beyond anxiety (e.g., excitement and joy), providing a more comprehensive understanding of the relationship between emotion and cognition.

In the future, the validity of the proposed physiological mechanism could be tested by comparing the processes presented in this study with human data. For instance, in a time estimation task, the model's predictions can be evaluated by collecting subjective ratings on time interval with varying anxiety levels and comparing them to simulation results. Additionally, verifying whether the model's predicted arousal changes align with actual physiological responses would be important. This could be achieved by combining electroencephalography (EEG) measurements with physiological indicators such as heart rate variability.

Furthermore, it is essential align the model's performance more closely with human performance. In this study, the model consistently underperformed compared to humans across all conditions. This discrepancy is likely due to differences in state transitions and operation methods between the model and human behavior (Nagashima, Nishikawa, Yoneda, Morita, & Junya, 2023). By adjusting the model to better match human performance, we can more accurately examine the impact of anxiety under conditions that more closely reflect human cognitive states.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP24KJ1215.

References

Anderson, J. R., Betts, S., Bothell, D., Hope, R., & Lebiere, C. (2019). Learning rapid and precise skills. *Psychological Review*, *126*(5), 727.

Bar-Haim, Y., Kerem, A., Lamy, D., & Zakay, D. (2010). When time slows down: the influence of threat on time perception in anxiety. *Cognition and Emotion*, *24*(2), 255–263.

Csikszentmihalyi, M. (1990). *Flow: The psychology of optimal experience*. New York: Harper & Row.

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.

Droit-Volet, S., & Meck, W. H. (2007). How emotions colour our perception of time. *Trends in Cognitive Sciences*, *11*(12), 504–513.

Fayolle, S., Gil, S., & Droit-Volet, S. (2015). Fear and time: fear speeds up the internal clock. *Behavioural Processes*, *120*, 135–140.

Gianferrara, P. G., Betts, S., & Anderson, J. R. (2021, 10). Cognitive & motor skill transfer across speeds: A video game study. *PLOS ONE*, *16*(10), 1-31.

Kotseruba, I., & Tsotsos, J. K. (2020). 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, *53*(1), 17–94.

Kujala, T., & Salvucci, D. D. (2015). Modeling visual sampling on in-car displays: The challenge of predicting safety-critical lapses of control. *International Journal of Human-Computer Studies*, *79*, 66–78.

LeDoux, J. E., & Pine, D. S. (2016). Using neuroscience to help understand fear and anxiety: a two-system framework. *American Journal of Psychiatry*, *173*(11), 1083–1093.

Maehigashi, A., Miwa, K., Terai, H., Kojima, K., & Morita, J. (2013). Experimental investigation of calibration and resolution in human-automation system interaction. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *96*(7), 1625–1636.

Matell, M. S., & Meck, W. H. (2000). Neuropsychological mechanisms of interval timing behavior. *Bioessays*, *22*(1), 94–103.

Morita, J., Miwa, K., Maehigashi, A., Terai, H., Kojima, K., & Ritter, F. E. (2020). Cognitive modeling of automation adaptation in a time critical task. *Frontiers in Psychology*, 2149.

Nagashima, K., Nishikawa, J., & Morita, J. (2024). Modeling task immersion based on goal activation mechanism. *Artificial Life and Robotics*, 1–16.

Nagashima, K., Nishikawa, J., Yoneda, R., Morita, & Junya. (2023). ACT-R Modeling of Rapid Motor Learning Based on Schema Construction. In *Proceedings of the 21th international conference on cognitive modeling.*

Nagashima, K., Nishikawa, J., Yoneda, R., Morita, J., & Terada, T. (2022). Modeling optimal arousal by integrating basic cognitive components. In *Proceedings of the 20th international conference on cognitive modeling.*

Rakitin, B. C., Gibbon, J., Penney, T. B., Malapani, C., Hinton, S. C., & Meck, W. H. (1998). Scalar expectancy theory and peak-interval timing in humans. *Journal of Experimental Psychology: Animal Behavior Processes*, *24*(1), 15.

Rosenbloom, P., Laird, J., Lebiere, C., Stocco, A., Granger, R., & Huyck, C. (2024). A proposal for extending the common model of cognition to emotion. In *Proceedings of the 22th international conference on cognitive modeling.* Tilburg, Netherlands.

Spielberger, C. D., Gonzalez-Reigosa, F., Martinez-Urrutia, A., Natalicio, L. F., & Natalicio, D. S. (1971). The state-trait anxiety inventory. *Revista Interamericana de Psicologia Interamerican Journal of Psychology*, *5*(3 & 4).

Stetson, C., Fiesta, M. P., & Eagleman, D. M. (2007). Does time really slow down during a frightening event? *PloS One*, *2*(12), e1295.

Taatgen, N. A., Van Rijn, H., & Anderson, J. (2007). An integrated theory of prospective time interval estimation: the role of cognition, attention, and learning. *Psychological Review*, *114*(3), 577.

Van Vugt, M. K., Van Der Velde, M., & Investigators, E.-M. (2018). How does rumination impact cognition? a first mechanistic model. *Topics in Cognitive Science*, *10*(1), 175–191.

Yerkes, R. M., & Dodson, J. D. (1908). The relation of strength of stimulus to rapidity of habit-formation. *Journal of Comparative Neurology and Psychology*, *18*(5), 459–482.

Zakay, D., Block, R. A., et al. (1995). An attentional-gate model of prospective time estimation. *Time and The Dynamic Control of Behavior*, *5*, 167–178.

Refining Memory Retrieval Models in a Dynamic Multi-Object Environment: Integrating Entropy and Observation Duration

Jongchan Pyeon (jbp5681@psu.edu)
Department of Industrial and Manufacturing Engineering
The Pennsylvania State University

Farnaz Tehranchi (farnaz.tehranchi@psu.edu)
School of Engineering Design and Innovation
The Pennsylvania State University

Abstract

Human memory retrieval in dynamic multi-object task environments is influenced by both cognitive processes and environmental structure. This study introduces a modified base-level learning equation that integrates entropy-based uncertainty and observation duration to better model retrieval behavior under such conditions. The approach was evaluated using behavioral data from 10 participants performing household tasks in the AI2-THOR simulated environment. Retrieval efficiency was measured through search time, defined as the interval between subtask inspection and interaction with the target object. Substantial variability in search time across participants reflected individual differences in memory access and action implementation. Compared to the conventional ACT-R base-level learning equation, the modified model yielded lower prediction error and closer alignment with observed behavioral trends. These findings underscore the potential value of integrating environmental structure into memory modeling and support the development of models that more accurately reflect human behavior in complex and dynamic task environments.

Keywords: Entropy; Cognitive Modeling; Learning; Memory Retrieval; ACT-R; AI2-THOR

Introduction

Simulating human cognition—including learning, decision-making, problem-solving, and perception—remains a significant challenge in artificial intelligence. Newell (1990) introduced the concept of unified theories of cognition, which involves acquiring knowledge, problem-solving, and perception. Cognitive architectures such as ACT-R (Anderson & Lebiere, 2014) and SOAR (Laird et al., 1987) provide structured frameworks for modeling these processes within unified systems (Kotseruba et al., 2016). Among these, ACT-R has emerged as one of the most influential architectures for representing cognitive functions such as memory, decision-making, and perception (Laird et al., 2017). It comprises programmable information processing mechanisms that model both interactions with the environment and internal cognitive operations (Byrne, 2012; Newell, 1990; Ritter, 2019). In ACT-R, knowledge is divided into two distinct types. Declarative knowledge consists of retrievable factual information stored in chunks, each defined by a set of attributes (slots) and their corresponding values. In contrast, procedural knowledge directs models' behavior through if-then production rules (Bothell, 2017).

A fundamental aspect of ACT-R is its base-level learning equation, which determines the activation value of memory chunks based on how frequently and recently they have been used (Anderson, 2007; Anderson & Lebiere, 1998). While effective in many contexts, this mechanism does not fully capture the adaptive nature of human memory, particularly in dynamic environments where multiple objects may appear, disappear, or change their locations. In such settings, retrieval demands extend beyond frequency and recency, requiring mechanisms that accommodate greater flexibility—the ability to adjust or shift strategies efficiently in response to varying conditions—and adaptability—the capacity to undergo lasting modifications to function effectively in dynamic environments—to more realistically reflect human cognition.

To address these limitations, researchers have extended ACT-R's memory retrieval mechanisms to better model human cognition. Stocco et al. (2010) introduced conditional routing for dynamic memory coordination, while Reitter et al. (2011) demonstrated how adaptive forgetting enhances flexibility. Anderson et al. (2004) integrated perceptual and motor processes for complex interactions, and Kim et al. (2013) explored skill acquisition across learning stages. Tehranchi et al. (2021) further examined the role of cognitive architectures in predicting learning and retention in complex task environment. Together, these studies highlight ACT-R's adaptability and reveal opportunities for further refinement in dynamic multi-object environments.

One crucial yet underexplored factor in memory retrieval is uncertainty, which influences how information is encoded, maintained, and recalled. Research shows that retrieval is more effective when uncertainty conditions during recall align with those present at learning, suggesting that memory systems are finely tuned to environmental unpredictability (Ogasa et al., 2024). Moreover, under high uncertainty, individuals shift from strict recall to adaptive learning, optimizing cognitive flexibility in dynamic contexts (Nicholas et al., 2022). Additionally, the brain actively tracks uncertainty in working memory, using it to regulate confidence in retrieved information (Yoo et al., 2021). These findings underscore that uncertainty is not merely a byproduct of memory retrieval but a fundamental mechanism that enhances flexibility, adaptability, and decision-making in complex dynamic environments.

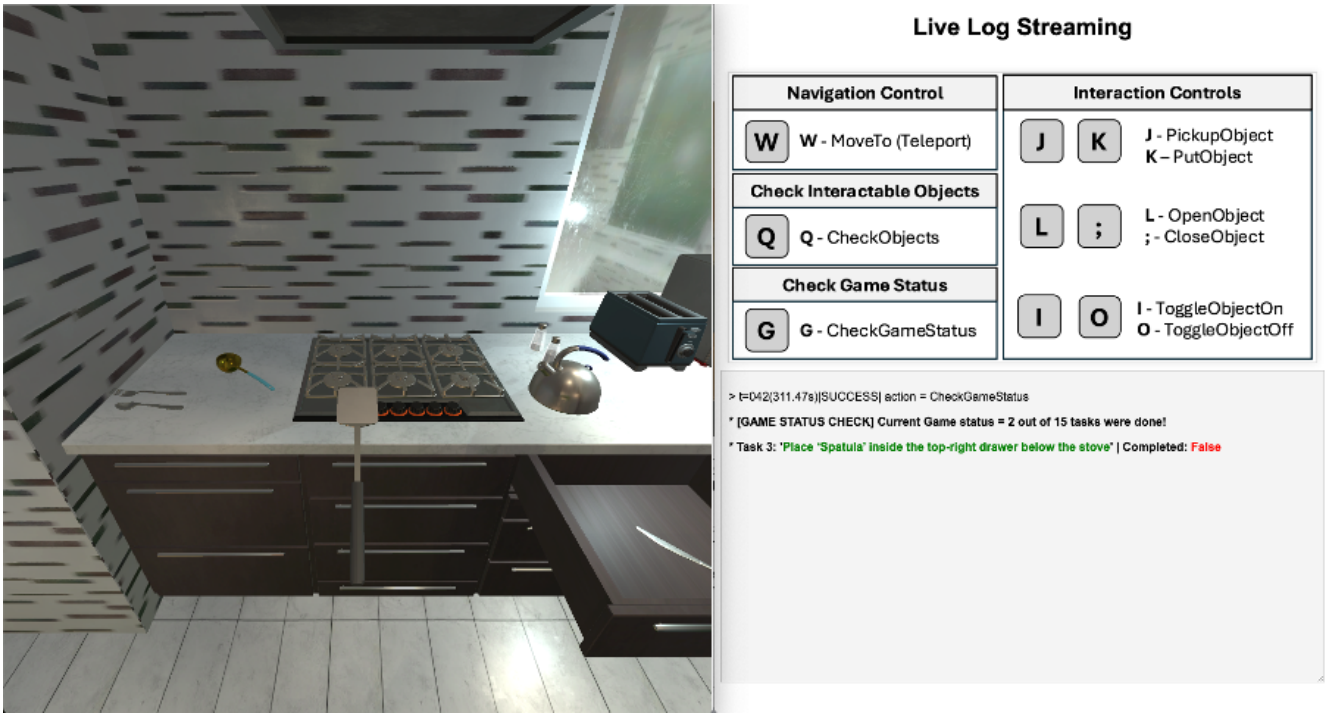


Figure 1. The user interface of the experiment: The left side on the screen displays the agent's current visual perspective within the AI2-THOR simulated environment and the right side on the screen shows the live log streaming window that provides real-time textual feedback on each executed action (e.g., indicating success, failure, cancellation, or wrong).

Furthermore, observation duration is critical in memory retrieval as it directly impacts encoding strength, consolidation, and recall efficiency. Research shows that retrieval success depends on the timing of memory reactivation, with shorter observation durations leading to weaker reconsolidation process and therefore increased forgetting over time (Inda et al., 2011). Retrieval practice within optimal time windows enhances long-term retention, indicating that insufficient observation time may hinder memory stabilization (Kriechbaum & Bäuml, 2023). Moreover, longer observation periods improve retrieval-related brain activity, reinforcing the idea that sustained exposure strengthens neural connections essential for recall (Bosshardt et al., 2005). These findings highlight that observation duration is not merely a passive factor but a crucial determinant of memory strength, influencing how effectively information is retained and accessed.

In dynamic multi-object environments, where objects may appear, disappear, or change locations, the concept of entropy offers a valuable measure for quantifying uncertainty. Introduced by Shannon (1948), entropy captures the amount of information required to describe the state of a variable, effectively representing environmental uncertainty (Thomas & Joy, 2006).

We hypothesize that memory retrieval in multi-object tasks is influenced not only by interaction frequency and recency, as described by the base-level learning equation, but also by object-related uncertainty and observation duration. By incorporating an entropy-based measure into ACT-R's base-level learning equation, we plan to develop a more accurate model that better accounts for variability in human behavior in memory retrieval in dynamic multi-object environments. To test this hypothesis, we designed a user experiment in which participants performed a series of household tasks within a dynamic multi-object environment using the AI2-THOR simulator (Kolve et al., 2017). This experiment examines how entropy and observation duration affect memory retrieval. Based on the findings, we modified the conventional base-level learning equation to more accurately simulate human behavior in memory retrieval within dynamic environments.

Method

This study aims to understand how the environmental uncertainty affects human memory retention in a dynamic multi-object environment. A series of household tasks was designed in the AI2THOR simulator. Using this environment, participants can interact with the objects in the map via the agent. This approach allows us to model human behavior in multi-object tasks through the analysis on the collected data (IRB approval number: STUDY00026274). The participants (n=10; Female: 2) are graduate students from The Pennsylvania State University.

Experiment Design

To assess our hypothesis, we designed a series of household tasks detailed in Table 1 and Table 2. The experiment is divided into two distinct phases: training (Table 1) and testing (Table 2), with completely different map configurations for each phase.

The user interface is divided into two primary sections. On the right side of the screen, the live log streaming window displays textual feedback for each executed action, indicating whether an action was successful, failed, canceled or wrong. On the left, the agent's view in the simulated AI2THOR environment provides the current visual perspective of the agent, as illustrated in Figure 1.

Participants have access to a total of 9 actions. For navigation, they can use the '*MoveTo*'. When the '*MoveTo*' button is pressed, a list of available locations appears, and participants select a destination by typing its two-digit index; the chosen destination is subsequently recorded in the live streaming log. For object interactions, six actions are available: '*PickupObject*', '*PutObject'*, '*OpenObject*', '*CloseObject*', '*ToggleOnObject*', and '*ToggleOffObject*'. When one of these buttons is pressed, a list of interactable objects at the current position is displayed, and the participant selects the desired object. To minimize keyboard stroke errors, the system prompts for confirmation with the message, "You want to select '<OBJECT/LOCATION>'? (Y/N)," and the action proceeds only upon receiving a 'Y' confirmation. Additionally, the '*CheckObject*' action provides information on the interactable objects at the current position, while the '*CheckGameStatus*' button allows participants to check the current task and its completion status. Once a task is completed, the next task is presented automatically, and pressing this button at the end of all tasks concludes the experiment.

This comprehensive interface and action set provides participants with the necessary tools to interact with the environment, laying the foundation for both the training and testing phases of the experiment. In the training phase, participants get hands-on experience controlling an agent in the AI2-THOR simulator by performing a series of tasks (Table 1). This phase is intended to help participants become comfortable with the actions needed in the simulator and prepares them to complete more complex tasks in the testing phase. The testing phase comprises three sequential tasks: *Exploration*, *Rearrangement*, and *Find & Place*.

Table 1. Task description of the training phase.

Task
(1) Place 'DishSponge' inside the right cabinet below the sink.
(2) Place 'Tomato' inside the fridge.
(3) Place 'Mug' inside the left cabinet next to the microwave.
(4) Place 'Knife' in the sink.
(5) Turn on the faucet.

In the *Exploration* task, participants explore the map by moving to all seven available locations to identify the interactable objects present at each location, as described in Table 2. After completing the *Exploration* task, participants take a one-minute break during which the agent's view is blocked to prevent them from obtaining any additional information. Once the break ends and the screen block is removed, they proceed directly to the *Rearrangement* task.

In the *Rearrangement* task, participants are asked to place objects at designated locations, completing 15 sub-tasks in order based on their observations from the *Exploration* phase and the instructions provided, as demonstrated in Table 2. After completing the *Rearrangement* task, participants take a three-minute break. Similar to the break after the *Exploration* task, the agent's view is blocked again to prevent them from obtaining any additional information. Once the break ends and the screen block is removed, they proceed directly to the *Find & Place* task. The breaks between each task are designed to induce memory decay.

Finally, the *Find & Place* task, which consists of eight sub-tasks (Table 2), challenges participants to complete a given task relying solely on their memory. This phase is designed to assess the efficacy of the base-level learning equation—employed in the ACT-R cognitive architecture—in simulating memory activation.

During both *Rearrangement* task and *Find & Place* task, unrelated actions to the current task are not executed. Instead, the users see a message of '*Wrong*'. All data related to executed actions, observed objects, and task completion times are collected for subsequent analysis.

Table 2. Task description of the testing phase. It is subdivided into the *Exploration*, *Rearrangement*, and *Find & Place* tasks.

Task	Subtask	
Exploration	(1-7) Explore the position 'Fridge', 'Sink', 'Table_1', 'Microwave & CoffeeMachine', 'Stove', 'Table_2', and 'Door' in order.	
Rearrangement	(1) Place 'Tomato' inside the microwave.	(9) Place 'DishSponge' inside the top-left drawer below the stove
	(2) Place 'Egg' inside the fridge.	
	(3) Place 'Spatula' inside the top-right drawer below the stove.	(10) Place 'Fork' inside the second drawer from the top at the center below the stove.
	(4) Place 'Kettle' inside the cabinet above the coffee machine	(11) Place 'Bowl' inside the cabinet below the coffee machine
	(5) Place 'Pot' on any of the stove burners.	(12) Place 'Knife' inside the second drawer from the bottom at the center below the stove.
	(6) Place 'Spoon' inside the bottom-right drawer below the stove.	(13) Find 'Cup' from the cabinet above the coffee machine and place it inside the left-most cabinet below the sink.
	(7) Place 'Ladle' inside the drawer below the countertop next to the sink.	(14) Place 'Pan' on any of the stove burners.
	(8) Place 'SoapBottle' inside the cabinet above the sink.	(15) Move to the position 'Door'
Find & Place	(1) Find 'Apple' and move it to the sinkbasin.	(5) Find 'Spoon' and move it to the sinkbasin.
	(2) Find 'ButterKnife' and move it to the sinkbasin.	(6) Find 'Ladle' and move it to the sinkbasin.
	(3) Find 'Fork' and move it to the sinkbasin	(7) Find 'DishSponge' and move it to the sinkbasin.
	(4) Find 'Bowl' and move it to the sinkbasin.	(8) Turn on the faucet.

User Study Procedure

The experiment was conducted in a controlled, quiet room with only the researcher and the participant present. Participants were seated in front of a 27-inch computer monitor and adjusted their seating to comfortably access the keyboard. They were briefed on the study's goals and procedures and were instructed to thoroughly read the on-screen instructions presented during the experiment. Participants were required to use only the keyboard to complete the tasks. The experiment began with the training phase, during which the researcher sat beside the participant to guide them through a series of tasks, facilitating their acclimation to the agent controls.

Before the testing phase began, participants were informed that if they selected an object that was not the target of the current task, they would receive a "Wrong" message, and the corresponding action would not be performed. As participants begin the *Exploration* task in the testing phase, they were informed to explore the position by pressing '*CheckObject*' button verbally once. Also, participants are informed to remember where they place objects once during the *Rearrangement* task. However, during the *Find & Place* task, no guides or no verbal instruction are provided to participants.

Base-level Learning Equations

The retrieval of memory is modeled by calculating the base-level activation of a memory chunk, which quantifies its accessibility based on both the frequency and recency of past presentations. In ACT-R, this is captured by the following equation:

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) \quad (1)$$

where t_j represents the time elapsed since the j^{th} presentation of the memory chunk, n is the total number of presentations, and d is a decay parameter (with values between 0 and 1) that determines how quickly the activation of the memory chunk decreases over time. In this study, t_j is measured by subtracting the past j^{th} observation time for object i during both the *Exploration* and *Rearrangement* tasks from the current time. The current time is defined as the moment when participants first checked the subtask containing the target object i during the *Find & Place* task. This formulation reflects the principle that memories become more accessible with more frequent or recent use, while their activation naturally diminishes as more time passes since the last encounter. The base-level learning equation is a cornerstone of the ACT-R cognitive architecture, as it provides a quantitative framework for modeling how memories are strengthened through repeated retrieval and gradually decay over time (Anderson, 2007; Anderson & Lebiere, 1998).

Building on these insights, our work extends the investigation by focusing on how both the number of interactable objects, which include openable and pickupable objects, encountered at each spatial position—quantified via entropy measures—and the duration of observation at each location affect memory retrieval. In dynamic environments, the standard base-level learning equation does not account for the variability in the amount of visual information available. To capture this additional source of uncertainty, we incorporate an entropy measure defined as:

$$H(X) = \log_2 n \quad (2)$$

where n is the number of openable objects (e.g., drawer, cabinet, fridge) at a given position (assuming a uniform probability distribution). Recognizing that attention may vary based on both the number of objects and the time spent observing a scene, we propose a modified base-level learning equation (B_i^H):

$$B_i^H = \ln\left[\sum_{j=1}^{n} \left(t_j^{-(d*H_j)}\right) * S_j\right] \quad (3)$$

In Equation (3), the method for measuring t_j, the elapsed time, is the same as in Equation (1). H, entropy, modulates the decay parameter d to reflect that a higher number of objects reduces the probability of retrieving any single memory. S_j represents the proportion of observation duration for object i that a participant spends in a j^{th} scene. Which means, the observation duration S_j is normalized by the number of pickupable objects in a j^{th} scene.

To compare the models' ability to predict search times, we applied the retrieval time equation from ACT-R (Bothell, 2017; Schunn & Anderson, 1998):

$$Retrieval\ Time = Fe^{-(f*A_i)} \quad (4)$$

Here, F is the latency factor, f is the latency exponent, and A_i represents the activation level of the chunk being retrieved. In ACT-R, A_i typically includes components such as base-level activation, spreading activation, partial matching, and noise. In this study, we consider only the base-level activation (B_i) as A_i to focus on memory strength as influenced by frequency, recency, entropy, and observation duration. We exclude spreading activation and partial matching, which are not consistently defined in dynamic multi-object environments. Both F and f were set to their default value of 1.

Results

The analysis was performed using time data collected for every action taken during the testing phase. To assess participants' performance, we measured search time, defined as the interval between the moment participants first checked the subtask and when they began interacting with the target object of the current subtask. For example, if a participant checked the content of a subtask at t = 100 seconds and then began interacting with the target object for that subtask at t = 180 seconds, the search time would be 80 seconds. Since every action taken during the testing phase was recorded along with timestamps and details, we were able to calculate the search time for each target object in each subtask during the *Find & Place* task in the testing phase.

Figure 2 presents the search time plotted for each subtask (i.e., target object) across all 10 participants. However, no consistent pattern emerged across participants, as illustrated in the figure. This variability in search time is influenced not only by differences in memory retrieval capacity but also by variations in action implementation time, as some participants may take longer to execute movements or interactions. To capture the general trend of participants' memory retrieval behavior during the household tasks, we calculated the mean of the search time for each subtask across all 10 participants.

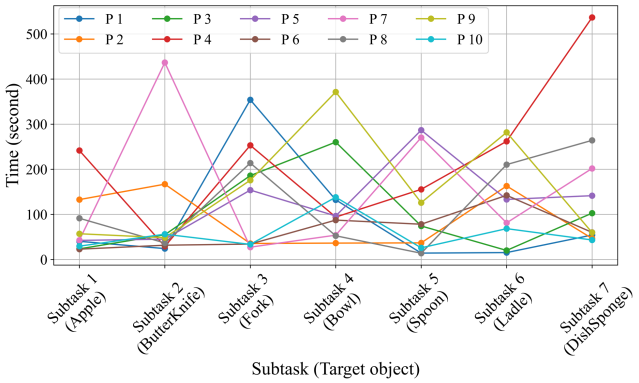


Figure 2. Search time plotted for each subtask (i.e., target object) in the *Find & Place* task across all participants.

To examine which equation, the conventional or the modified, better reflects participants' search behavior, the search time for each subtask was averaged across all participants to observe a generalized trend (Figure 3a). Then, the base-level activation values for each subtask were computed using both the conventional equation (Equation 1) and the modified equation (Equation 3) and then averaged

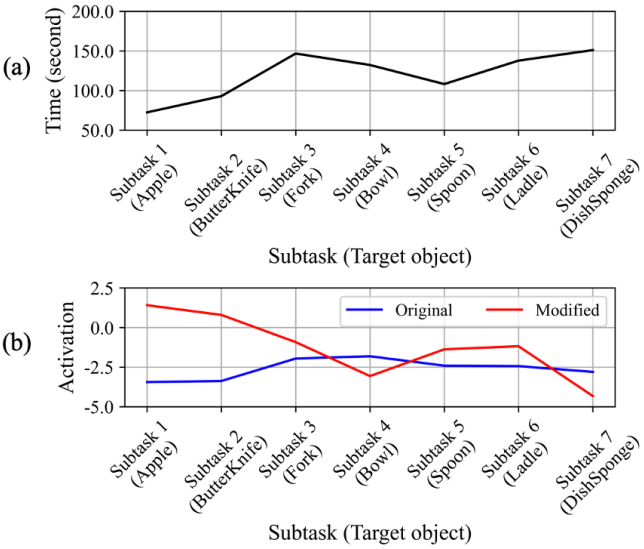


Figure 3. Average search times and predicted activation values across subtasks. (a) Average search time per target object. (b) Averaged base-level activation values computed using the conventional (Eq. 1, blue) and modified (Eq. 3, red) equations. All values are averaged across participants to show general trends.

across all participants to observe generalized trends in predicted retrieval performance (Figure 3b). For both equations, the decay parameter d was set to its default value of 0.5. Because search time reflects retrieval efficiency, it can be interpreted through base-level activation, which is inversely proportion to search time. Higher activation corresponds to faster retrieval, meaning shorter search times. In Figure 3a, the black line shows average search time across subtasks. In Figure 3b, the red line, representing the modified equation, aligns more closely with the search time trend than the conventional equation (blue), suggesting better modeling of retrieval dynamics.

In addition to this qualitative trend comparison, we conducted a quantitative evaluation of each model's predictive performance. Predicted retrieval times were computed using Equation 4 based on the calculated base-level activation values. Retrieval times were normalized using min-max scaling for the MAE calculation, as no parameter optimization was applied to either the original or modified equation. We assessed the accuracy of these predictions using Mean Absolute Error (MAE) and examined how well each model captured monotonic trends using Spearman correlation coefficients (with p-values), computed for each participant. The results were then averaged across participants (see Table 2). The original equation yielded an average MAE of 0.45 and a negative average Spearman correlation of –0.22. In contrast, the modified equation achieved a lower average MAE of 0.35 and a higher positive average correlation of 0.30.

Table 3. MAE and Spearman correlation (with p-value) for predicted retrieval time using the original and the modified equations.

Participant	MAE		Corr (p-value)	
	Original	Modified	Original	Modified
P1	0.57	0.39	-0.86 (0.01)	0.54 (0.22)
P2	0.49	0.33	-0.29 (0.53)	0.29 (0.53)
P3	0.27	0.36	0.29 (0.53)	0.21 (0.64)
P4	0.20	0.55	0.86 (0.01)	0.39 (0.38)
P5	0.31	0.24	0.29 (0.53)	0.32 (0.48)
P6	0.51	0.32	-0.39 (0.38)	0.11 (0.82)
P7	0.52	0.35	-0.57 (0.18)	0.54 (0.22)
P8	0.55	0.33	-0.64 (0.12)	0.68 (0.09)
P9	0.53	0.32	-0.04 (0.94)	0.21 (0.64)
P10	0.56	0.34	-0.82 (0.02)	0.50 (0.25)
Average	0.45	0.35	-0.22	0.30

Discussion

To model memory retrieval in dynamic task environments, we applied both the conventional base-level learning equation (Equation 1) and a modified version (Equation 3) that incorporates entropy-based uncertainty and observation duration. Since search time reflects retrieval efficiency, it can be interpreted through base-level activation: stronger memory traces result in shorter search times, while weaker memory leads to longer ones (Anderson, 2007; Anderson & Lebiere, 1998; Bothell, 2017).

In the qualitative analysis, we compared trends in average search times (Figure 3a) with activation values predicted by

both equations (Figure 3b). The modified equation produced activation patterns that more closely aligned with observed search time fluctuations. Specifically, it captured decreases in activation where search time increased, a pattern not consistently reflected by the conventional equation. This suggests that incorporating entropy and observation duration improves the model's sensitivity to dynamic memory conditions in multi-object environments.

To quantitatively evaluate model performance, we used ACT-R's retrieval time function (Equation 4) to convert activation values into predicted retrieval times. As shown in Table 3, the modified equation outperformed the conventional model for most participants in terms of Mean Absolute Error (MAE). The average MAE across participants decreased from 0.45 (original) to 0.35 (modified), indicating improved prediction accuracy. Additionally, the modified model yielded more positive Spearman correlation coefficients with observed search times for 8 out of 10 participants, whereas the conventional model produced mostly negative or near-zero correlations.

Although most of the modified model's correlations were not statistically significant, they were directionally consistent with the observed data. For instance, participants P1, P7, and P8 showed substantial improvements in correlation values, shifting from strong negative correlations under the conventional model to moderate positive values with the modified model. Participant P10 notably exhibited a significant negative correlation with the original model (r = –0.82, p = 0.02), which became positive with the modified model (r = 0.50), though it was no longer statistically significant. Overall, the modified model correctly captured the trend direction, as participants with higher actual search times also tended to have higher predicted times. However, the strength of these associations was generally modest.

The lack of statistical significance in most cases is likely due to the limited number of task observations (n = 8) per participant, which reduces statistical power. Nevertheless, the consistent reduction in MAE and the shift toward positive correlations indicate that the modified model better reflects the directionality of memory-based task performance. Optimizing model parameters was not performed in this study, and the current results were obtained using default values (e.g., decay rate, latency factor and exponent), which may have limited the model's ability to fully capture individual differences in retrieval behavior.

Despite the improvements introduced by the modified base-level learning equation, several limitations should be acknowledged. First, the model does not account for individual differences in memory retrieval strategies, attention allocation, or navigation behavior. Although search time was used as a proxy for retrieval efficiency, it may also reflect extraneous factors such as motor variability, navigation speed, or exploratory behavior. As a result, some variation in search time and predicted activation may stem from cognitive or behavioral differences that are not captured by entropy and observation duration. Second, the method for estimating observation duration assumes that participants distribute their attention equally across all pickupable objects in a scene. This assumption may not hold in practice, as participants likely focus more on task-relevant or visually salient objects. Without capturing this unequal attention allocation, the model may misrepresent the actual observation dynamics influencing memory retrieval. Third, the entropy calculation is based on a uniform probability distribution of object occurrences across spatial locations. However, participant behavior suggests a more structured search strategy. Rather than sampling the environment randomly, participants tend to inspect closed receptacles when uncertain, guided by spatial configuration, salience, and task demands. This simplification may underestimate the complexity of human search behavior in realistic environments. Fourth, the parameters in the base-level learning and retrieval time equations were kept constant across all participants. In practice, these parameters may vary depending on individual characteristics or task conditions. The absence of parameter optimization may have limited the model's ability to capture variability in retrieval performance and prediction accuracy.

Future work should address these limitations by incorporating individualized model parameters and refining them through data-driven optimization techniques. Adjusting key parameters, such as the decay rate in the base-level learning equation, based on task context or individual behavior could enhance the model's adaptability and predictive precision. Improved methods for estimating observation duration, such as using eye-tracking to measure unequal attention allocation, may provide a more accurate representation of perceptual input. Additionally, replacing the uniform entropy assumption with a context-sensitive model that accounts for object relevance and spatial layout may yield a more realistic understanding of search and memory processes in complex environments. Together, these enhancements would contribute to a more robust and interpretable model of human memory retrieval in dynamic task settings.

Conclusion

This study demonstrates a modified base-level learning equation that incorporates entropy-based uncertainty and observation duration to improve the modeling of memory retrieval in dynamic multi-object environments. Applied to behavioral data from a simulated household task setting using the AI2-THOR environment, the modified equation produced lower prediction errors and more consistent alignment with observed search time trends compared to the conventional ACT-R's base-level learning equation. The results suggest that accounting for perceptual exposure and environmental uncertainty can enhance the sensitivity of cognitive models to retrieval dynamics in realistic task contexts. These findings underscore the potential value of integrating environmental structure into memory modeling and contribute to a more realistic understanding of human behavior in complex dynamic task environments.

References

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

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

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

Anderson, J. R., & Lebiere, C. J. (2014). *The atomic components of thought*. Psychology Press.

Bosshardt, S., Degonda, N., Schmidt, C. F., Boesiger, P., Nitsch, R. M., Hock, C., & Henke, K. (2005). One month of human memory consolidation enhances retrieval-related hippocampal activity. *Hippocampus*, *15*(8), 1026-1040.

Bothell, D. (2017). ACT-R 7 Reference Manual. act-r.psy.cmu.edu/wordpress/wp-content/themes/ACT-R/actr7/reference-manual.pdf

Byrne, M. D. (2012). Unified theories of cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, *3*(4), 431-438.

Inda, M. C., Muravieva, E. V., & Alberini, C. M. (2011). Memory retrieval and the passage of time: from reconsolidation and strengthening to extinction. *Journal of Neuroscience*, *31*(5), 1635-1643.

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.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Deitke, M., Ehsani, K., Gordon, D., & Zhu, Y. (2017). Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Kotseruba, I., Gonzalez, O. J. A., & Tsotsos, J. K. (2016). A review of 40 years of cognitive architecture research: Focus on perception, attention, learning and applications. *arXiv preprint arXiv:1610.08602*.

Kriechbaum, V. M., & Bäuml, K.-H. T. (2023). The critical importance of timing of retrieval practice for the fate of nonretrieved memories. *Scientific reports*, *13*(1), 6128.

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.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*(1), 1-64.

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press.

Nicholas, J., Daw, N. D., & Shohamy, D. (2022). Uncertainty alters the balance between incremental learning and episodic memory. *Elife*, *11*, e81679.

Ogasa, K., Yokoi, A., Okazawa, G., Nishigaki, M., Hirashima, M., & Hagura, N. (2024). Decision uncertainty as a context for motor memory. *Nature Human Behaviour*, *8*(9), 1738-1751.

Reitter, D., Keller, F., & Moore, J. D. (2011). A computational cognitive model of syntactic priming. *Cognitive Science*, *35*(4), 587–637.

Ritter, F. E. (2019). Modeling human cognitive behavior for system design. In *S. Scataglini and G. Paul (Eds.), DHM and posturography* (pp. 517-525).

Schunn, C. D., & Anderson, J. R. (1998). Scientific discovery. In J. R. Anderson & C. Lebiere (Eds.), *The atomic components of thought*. Erlbaum.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, *27*(July & October), 379–423 & 623–656.

Stocco, A., Lebiere, C., & Anderson, J. R. (2010). Conditional routing of information to the cortex: A model of the basal ganglia's role in cognitive coordination. *Psychological Review*, *117*(2), 541-574.

Tehranchi, F., Oury, J. D., & Ritter, F. E. (2021). Predicting learning and retention of a complex task using a cognitive architecture. Proceedings of the Annual Meeting of the Cognitive Science Society, 43, 1077-1083.

Thomas, M., & Joy, A. T. (2006). *Elements of information theory*. Wiley-Interscience.

Yoo, A. H., Acerbi, L., & Ma, W. J. (2021). Uncertainty is maintained and used in working memory. *Journal of vision*, *21*(8), 13.

Proceedings of the 23rd International Conference on Cognitive Modelling (ICCM 2025)

Active Inference and Psychology of Perception: A Critical Study

Dhanaraaj Raghuveer (dhanaraaj.raghuveer@uni-marburg.de)
Department of Psychology, Philipps Universität Marburg
Gutenbergstraße 18, 35032 Marburg

Dominik Endres (dominik.endres@uni-marburg.de)
Department of Psychology, Philipps Universität Marburg
Gutenbergstraße 18, 35032 Marburg

Abstract

Active Inference is an ambitious research program in cognitive science and mathematical psychology aiming to provide a unified account of perception, goal-oriented action, and cognition. In this analytical and critical study we argue for the inadequacy and incompleteness of active inference's theory of perception. Specifically, we show that 'perception as inference,' considered as an explanatory psychological/philosophical theory of perception, faces the 'problem of prior knowledge' - the problem of accounting for the origin and nature of knowledge assumed in the Bayesian agent's generative model. We trace the conceptual inadequacies to the psychology/philosophy of Helmholtz and adumbrate an alternate theory of direct perception based on the psychology/philosophy of Whitehead.

Keywords: Active Inference; Bayesian modeling; Perception as Inference; Indirect and Direct Perception; Mathematical and Philosophical Psychology.

Introduction

Active inference is one of the most prominent theories in contemporary cognitive science and mathematical psychology. It goes by several names in the cognitive science literature, such as Bayesian Brain, Predictive Processing, Predictive Coding, and Free Energy Principle - all related by a core set of ideas but with slight variations. One distinguishing characteristic of active inference is that it gives a unified account of perception, goal-oriented action, and cognition (Parr, Pezzulo, & Friston, 2022). Beyond the domains mentioned above of psychology, it has also been applied in neuroscience (Friston, 2010, 2009), neurophilosophy (Clark, 2013; Hohwy, 2014), higher cognition (Pezzulo, Rigoli, & Friston, 2015, 2018), and robotics (Oliver, Lanillos, & Cheng, 2022). Furthermore, it is also claimed to bear relevance beyond cognitive science to life science (Badcock, Ramstead, Sheikhbahaee, & Constant, 2022; Ramstead, Badcock, & Friston, 2018) and physical science (Fields et al., 2023a, 2023b).

Such wide-ranging applications indicate the importance of studying the theory analytically and critically for cognitive science. Such an analytical and critical study is what this paper is. The paper's main argument is that Active Inference, as it is currently formulated, is inadequate and incomplete in the domain of perceptual experience. To defend this argument, we must first elucidate what active inference is, what its theory of perception is, and what is inadequate/incomplete about this theory.

The following sections successively aim to undertake this elucidation. In the next section, we give a brief mathematical introduction to active inference, specifically what is called the 'Low Road' (Parr et al., 2022). This section is based on Biehl, Guckelsberger, Salge, Smith, and Polani (2018) owing to their clarity and explicit assertion of their modeling assumptions. However, our conceptual clarification and criticism apply equally to *all* active inference works, including Biehl et al. (2018).

The subsequent section, titled **"Perception as Inference"**, elucidates the psychological import and the conceptual inadequacies of active inference's theory of perception. The crux of this section comes down to the differentiation between a psychological/philosophical theory of perception as inference and the engineering/robotics simulation of it. The former faces what we call the 'problem of prior knowledge' while it is addressable through the engineer/roboticist's perspective for the latter. We end the paper with the adumbration of an alternate theory of direct perception in contrast to the inferential & indirect perception of active inference and a brief conclusion.

'Low Road' to Active Inference

The best way to introduce the 'Low Road' is by treating it as a form of generalized reinforcement learning (Millidge, Tschantz, & Buckley, 2021). Most existing simulations and implementations of active inference are carried within the Partially Observable Markov Decision Process (POMDP) framework, whose relation to RL literature is well known.

In this generalized RL setup, there exists an agent-environment process, which is the complete dynamics underlying the agent, environment, and their interactions. One could formalize this complete agent-environment system dynamics as a causal Bayesian network (see Fig 1). The random variables E, S, I, A stand respectively for external states, sensory states, internal states & active states, and we depict their respective state spaces as $\mathcal{E}, \mathcal{S}, I$, and $\mathcal{A}$. The causal dependencies depicted in Fig 1 tell the following story: The initial external state ($e_0 \in \mathcal{E}$) generates a sensory state ($s_0 \in \mathcal{S}$), which in turn influences the internal state at first time step ($i_1 \in I$). Conditional on this internal state, an action ($a_1 \in \mathcal{A}$) is performed, which together with the previous external state yields a new external state ($e_1 \in \mathcal{E}$).

This new external state produces a sensory state, and the cycle continues until the final time step T (which is 3 in our case). This, in a nutshell, is the perception-action cycle ac-

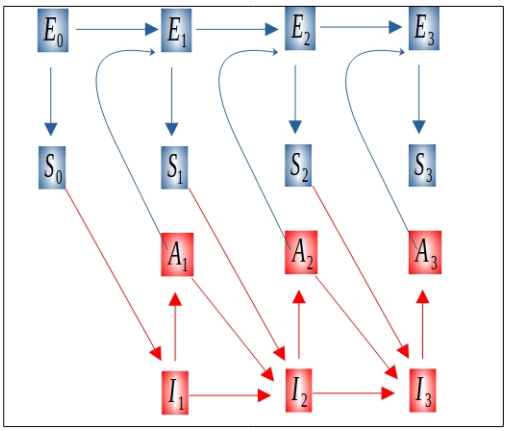


Figure 1: Bayesian network depiction of the complete dynamics underlying the agent-environment system. The red boxes correspond to aspects of the agent (dynamics), while the blue boxes correspond to the environment (dynamics). See text for more description.

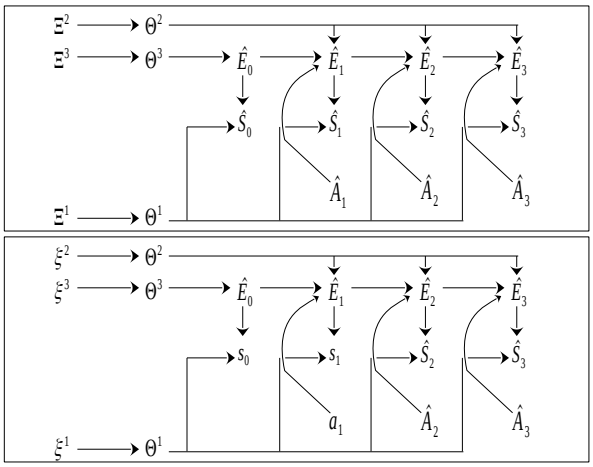


Figure 2: **Top**: Bayesian network depiction of the agent's generative model with the inaccessibility and internal state 'conditions' applied when no data is available. **Bottom**: The same Bayesian network with plugged-in data until $t = 2$ and an additional assumption of fixed hyperparameters. See text for further description. Adapted from Biehl et al. (2018).

cording to active inference and generalized RL. The dependency relation from A_t, I_t to I_{t+1} represents that the agent's internal state keeps track of all its previous active, internal, and sensory states. In this sense, one important aspect of the internal state at any time step t is that it is a sequence of all sensory states and active states until $t-1$. That is, $i_t = (s_0, s_1 a_1, s_2 a_2, ..., s_{t-1} a_{t-1})$, which we succinctly represent as $i_t = sa_{\prec t}$.

Given this Bayesian network formalization of the agent-environment process, one can now define the joint probability distribution of the complete dynamics as:

$$p(e_{0:T}, s_{0:T}, a_{1:T}, i_{1:T}) = p(e_0)p(s_0|e_0) \left(\prod_{t=1}^{T} p(i_t|a_{t-1}, s_{t-1}, i_{t-1}) p(a_t|i_t) \times p(e_t|e_{t-1}, a_t) p(s_t|e_t) \right) \quad (1)$$

with the assumption that $p(i_1|s_0, a_0, i_0) = p(i_1|s_0)$ and $e_{0:T}$ denotes $(e_0, e_1, ..., e_T)$. To fully determine this joint probability distribution, one has to specify the underlying state spaces $\mathcal{E}, \mathcal{S}, I, \mathcal{A}$ and all the factors of the above equation. Of special importance are - external dynamics $p(e_t|e_{t-1}, a_t)$, sensory dynamics $p(s_t|e_t)$, and action generation $p(a_t|i_t)$. The former two are important aspects of the environment, while the latter is an important aspect of the agent in this agent-environment process.

The ultimate aim of an active inference agent is to perform appropriate action $a_t \in \mathcal{A}$ through its action generation mechanism. At this action generation phase of the complete agent-environment dynamics, all the critical features of active inference get cashed out. By way of anticipating, the four key features are - 1) Inference, 2) Approximation, 3) Evaluation, and 4) Selection. The end product of active inference is that the agent *selects* actions that are *appropriate* using *approximate* Bayesian *inference*. We will focus here only on the *inference* feature as this is the only relevant feature for this paper.

Inference

Within the action generation mechanism $p(a_t|i_t)$ of active inference, it is assumed that the complete dynamics of the agent-environment system, *generative process*, is hidden from the agent - "true environmental contingencies, or the generative process, which is inaccessible to the organism" (Parr et al., 2022). However, some aspects of the complete dynamics like $\mathcal{S}, \mathcal{A}$ are explicitly assumed to be accessible.[1] Combining this inaccessibility with our specification above of $i_t = sa_{\prec t}$, we nearly get what is called the *generative model* of the agent.

Specifically, what the generative model does is that it tries to model all the random variables of the complete dynamics through estimated variables $\hat{E}, \hat{S}$, and $\hat{A}$. As mentioned above, it replaces the internal state with the sequence of past sensory and action values wherever necessary. Furthermore, the internal state is also cast as additional parameters $(\Theta^1, \Theta^2, \Theta^3)$ & hyperparameters (Ξ^1, Ξ^2, Ξ^3) which track specific aspects of the environment dynamics and are also considered as random variables. The combined result is the generative model,

[1] We would like to note here that the random variable S is classified as part of the environment because the sensory dynamics which generates the next sensory state is classified as part of the environment dynamics. This assumption of accessibility is valid because S is the sensory *interface* between the agent and environment. The same goes for A, which is the action *interface*.

which can also be formalized as a causal Bayesian network (see Fig 2 **Top**).

Just as in the previous case of complete dynamics, one can now define the joint probability distribution of the generative model as:

$$q(\hat{e}_{0:T},\hat{s}_{0:T},\hat{a}_{1:T},\theta^1,\theta^2,\theta^3,\xi^1,\xi^2,\xi^3) = q(\hat{e}_0|\theta^3)q(\hat{s}_0|\hat{e}_0,\theta^1) \times\left(\prod_{t=1}^{T} q(\hat{e}_t|\hat{e}_{t-1},\hat{a}_t,\theta^2)q(\hat{s}_t|\hat{e}_t,\theta^1)q(\hat{a}_t)\right) \times\prod_{i=1}^{3} q(\theta^i|\xi^i)q(\xi^i) \tag{2}$$

To completely determine this joint probability distribution, one has to specify the underlying state spaces $\hat{\mathcal{E}},\hat{\mathcal{S}},\hat{\mathcal{A}}$ and the state spaces of parameters (Δ_{Θ^i}) and hyperparameters (Δ_{Ξ^i}). It is explicitly assumed that the agent has access to the complete dynamics' sensory and active state spaces. Thus, $\hat{\mathcal{S}}=\mathcal{S}$ and $\hat{\mathcal{A}}=\mathcal{A}$. However, certain complications arise regarding the state space of $\hat{E}$ and Θ^i, which will be important for us later. We will simply note it here. With respect to the hyper parameters Ξ^1,Ξ^2, and Ξ^3 each of them are fixed to $\xi^1\in\Delta_{\Xi^1}$, $\xi^2\in\Delta_{\Xi^2}$, $\xi^3\in\Delta_{\Xi^3}$ respectively.

Equipped with this generative model and its assumptions, we now move on to the 'ultimate aim' of active inference agents. To perform appropriate actions, the agent must first consider the data it currently has. To be precise, at any time step t during which the action selection mechanism is engaged, the agent's generative model incorporates the available data until that time step. This is depicted in Fig 2 **Bottom**. Given this plugging in of data, one can get the posterior probability distribution on the non-observed variables as:

$$q(\hat{e}_{0:T},\hat{s}_{t:T},\hat{a}_{t:T},\theta|sa_{\prec t},\xi) = \frac{q(\hat{e}_{0:T},\hat{s}_{t:T},\hat{a}_{t:T},sa_{\prec t},\theta,\xi)}{\int\sum_{\hat{e}_{0:T},\hat{s}_{t:T},\hat{a}_{t:T}} q(\hat{e}_{0:T},\hat{s}_{t:T},\hat{a}_{t:T},sa_{\prec t},\theta,\xi)d\theta} \tag{3}$$

where θ denotes $(\theta^1,\theta^2,\theta^3)$ in short handed manner. The same goes for ξ. In comparison to Eq.2, there are no major changes here, except that the available data ($sa_{\prec t}$) collapses $\hat{s}_{0:t-1}$ to $s_{\prec t}$ and $\hat{a}_{1:t-1}$ to $a_{\prec t}$. The rest is just the Product rule applied to the joint distribution and the appropriate marginalization.

An additional facet of active inference agents is that $sa_{\prec t}$ is not their only available data. Specifically, they consider 'counterfactual' future actions as given data and derive the posterior probability conditional on these 'counterfactual' actions. "Active Inference assumes that agents use a (generative) model to [...] evaluate their future (and *counterfactual*) action possibilities."(Parr et al., 2022, emphasis ours). What these counterfactual future actions mean is that the agent now considers multiple alternative future action sequences $\hat{a}^1_{t:T}$, $\hat{a}^2_{t:T}$, ..., and for each such action sequence one posterior probability is associated. Formally,

$$q(\hat{e}_{0:T},\hat{s}_{t:T},\theta|\hat{a}_{t:T},sa_{\prec t},\xi) = \frac{q(\hat{e}_{0:T},\hat{s}_{t:T},\hat{a}_{t:T},sa_{\prec t},\theta,\xi)}{\int\sum_{\hat{e}_{0:T},\hat{s}_{t:T}} q(\hat{e}_{0:T},\hat{s}_{t:T},\hat{a}_{t:T},sa_{\prec t},\theta,\xi)d\theta} \tag{4}$$

The above equation is critical in understanding active inference and is called *Bayesian complete posterior*. At this stage in our journey, a quick recap would be helpful. We first started with our agent-environment process. Within that process lies an important aspect of the agent dynamics called the action generation mechanism. In the action generation mechanism, it is assumed that the agent only has partial access to the complete dynamics. Thus, a generative model is posited with some assumptions and conditions that reflect this partial accessibility. Furthermore, the agent can derive a posterior distribution on its non-observed variables by considering the available and 'counterfactual' data.

What the agent now has in its possession is a set of complete posteriors $\{q(\hat{e}_{0:T},\hat{s}_{t:T},\theta|\hat{a}^1_{t:T},sa_{\prec t},\xi),q(\hat{e}_{0:T},\hat{s}_{t:T},\theta|\hat{a}^2_{t:T},sa_{\prec t},\xi),...\}$, each one corresponding to one future action sequence. This set encapsulates the complete knowledge of the agent at that time step. This complete knowledge constitutes the first of the four key features of active inference mentioned above - Inference.

Put simply, the agent has *inferred* using Bayesian principles, knowledge about the hidden external states, model parameters, and future sensory states. Suppose that the agent is endowed with a specific objective (like, in RL, to maximize rewards). In that case, the agent can *evaluate* $\{\hat{a}^1_{t:T},\hat{a}^2_{t:T},...\}$ and rank according to this normative criteria, and then *select* actions appropriately. However, for that evaluation to happen, it should first be considered whether this set of complete posteriors can be tractably computed. That is where the *approximate* part of active inference comes in (which we will not get into).

Conceptually, what Eq. 4 means is that the agent gains knowledge about the present and past external world ($\hat{e}_{0:t}$) through indirect Bayesian inference.[2] It is this posterior inference of the present and past external world that leads to active inference's claim of *perception* as inference.

This completes our brief mathematical introduction to active inference. As mentioned above, there are other key features like approximation, evaluation, and selection that we have not gotten into. For the purposes of this paper, we have abstracted from those features and highlighted the relevant aspects of active inference's theory of perception. In the next section, we will look deeper into 'perception as inference' and what difficulties we find in it as a psychological theory.

[2]Correlative to perception as inference there is also the 'action as inference' or 'planning as inference' (Parr et al., 2022) which is computed during the *evaluation* phase. Even here, only future active states are evaluated while past active states are *given* and *directly* accessible in contrast to external states, which are *inferred* irrespective of whether they are past, present, or future. In other words, it is assumed that the agent is cut off from its external world at all times.

Perception as Inference

In this section, we will start by conceding to all the explicit assumptions of active inference like Markovian assumptions, factorization assumptions, idealized memory in i_t, accessibility of $\mathcal{S}, \mathcal{A}$ and inaccessibility of $\mathcal{E}$ to the agent, the biological plausibility of computing Eq. 4 for all possible action sequences, and so on. However, we will challenge one of the core claims of active inference - *perception* as inference (Parr et al., 2022) - which is supposed to follow from these explicit assumptions. Specifically, perception as inference is supposed to follow from the assumption that in the perception-action cycle, the agent is cut off from its environment ($\mathcal{E}$ and E are inaccessible to the agent) and thus has to infer the external world through a model ($\hat{\mathcal{E}}, \hat{E}$).

Given this explicit assumption that an agent is cut off from its environment, the question then becomes how does an active inference agent ever get started on its $\hat{\mathcal{E}}$? What dimensionality to that state space does it assign? What nature or quality does it assume holds for that space? If it assumes a continuous nature, then why is it continuous? If discrete, then why so? What sort of metric or measure holds in that space? If the distribution over that space is assumed to be categorical, then why is that? What is the cardinality of that space? What are the categories making up that categorical distribution? More importantly, what is the source of all these assumptions and knowledge? We argue that active inference answers all these questions by implicitly 'smuggling' knowledge. We will elaborate and provide evidence for what we mean by implicit 'smuggling.'

First and foremost, the cardinality of $\mathcal{E}$ and $\hat{\mathcal{E}}$ are assumed to be the same, i.e. $|\mathcal{E}| = |\hat{\mathcal{E}}|$. "Note that we are not inferring the causal structure of the Bayesian network or state space cardinalities, but define the generative model as a fixed Bayesian network [...]" (Biehl et al., 2018). While the question of how the agent comes up with the causal structure is important, we will focus here on the cardinality. This is the first example of implicit 'smuggling' of knowledge, all the while holding that the external dynamics and its associated properties are "inaccessible to the organism"(Parr et al., 2022).

Secondly, the structure, nature, and categories of $\hat{\mathcal{E}}$ are also supposed to be the same as the structure of $\mathcal{E}$. "All that matters is that the environment accepts the agent's actions and returns observations that are discrete and *are compatible with the support of the likelihood* [$q(\hat{s}_\tau|\hat{e}_\tau, \theta)$ in our notation] of the agent's generative model."(Heins et al., 2022, emphasis ours). In other words, $\hat{E}/\hat{\mathcal{E}}$ are implicitly assumed to be 'compatible' with $E/\mathcal{E}$ so that the likelihood distributions are 'compatible' (Given that we have already assumed $S/\mathcal{S}$ and $\hat{S}/\hat{\mathcal{S}}$ have the same structure and categories). This is another instance of implicit 'smuggling' of knowledge into the agent.

There are many examples of such implicit 'smuggling' along with the explicit assumption of 'inaccessibility' simultaneously. For example, in the same Biehl et al. (2018) paper, they say "the [model's] environment dynamics [...] are not inferred and are [...] *set to the physical environment dynamics*: $q(\hat{e}_\tau|\hat{e}_{\tau-1}, \hat{a}_\tau, \theta^2) = p(e_\tau|e_{\tau-1}, a_\tau)$."

They mention the same for sensor dynamics: "Similarly, since the sensor dynamics [..] are also not inferred, we find: $q(\hat{s}_\tau|\hat{e}_\tau, \theta^1) = p(s_\tau|e_\tau)$." Finally, if one looks at the Appendix of the same paper where they translate their clear & consistent notation to the notations used in the main active inference papers, one finds that no distinction is even made between $e_t \in \mathcal{E}$ and $\hat{e}_t \in \hat{\mathcal{E}}$ in the latter.[3] They are implicitly assumed to be the same!

In sum, the supposed 'inaccessibility' of the external dynamics does not exist in any substantial sense. All the psychologically relevant aspects, like the dimensionality, quality, structure, and categories of the external world, are already 'smuggled' into the agent's model. The only sense in which the external states are 'inaccessible' to the agent is the *probability values*. There is only an inference of *probabilities* here, but not the categories and spaces over which the probabilities are defined.

For a concrete example, in the classic two-armed bandit task, the categories 'LEFT' and 'RIGHT' exist as external states. There is no sense in which these states are 'hidden' for the same categories 'LEFT' and 'RIGHT' exist in the generative model. What is 'hidden' is only the values $(0.8, 0.2)$, and it is only these probability values that are inferred and updated in a rational Bayesian manner (say to $(0.6, 0.4)$). The conceptually and psychologically relevant part is how humans come to recognize those categories & structures and how perception contributes to gaining such natural knowledge. Active inference does not address that issue, and this set of arguments depicts our challenge to the theory's core claim - *perception* as inference.

Let us state what we are *not* claiming. We are not claiming that it is improper or objectionable to use the generalized RL framework or active inference in the context of an external engineer or a designer to explicitly design a robot or a machine for a fixed or artificial environment. In this case, the source of all the assumptions and 'smuggling' comes directly from the external human's knowledge. However, when one moves from this technological & engineering context to the scientific & philosophical context of explaining human perception itself as such a process, the 'smuggled' *prior knowledge* is unaccounted for. It is this move that is problematic.

Having laid down this background, it is appropriate to ask why or whose authority active inference theorists claim for this problematic move from a potentially useful (but limited) technological context to the explanatory scientific and philosophic context. It is well known in the literature that the authority of 19th-century polymath Hermann von Helmholtz is claimed to justify this move. Specifically, his theory of 'perception as unconscious inference.' Helmholtz is invoked four times in Parr et al. (2022) w.r.t perception, but at no place do they engage with the psychological and philosophical import

[3] Such clarity, consistency, and explicit assertions are a few reasons why we relied on Biehl et al. (2018) for our mathematical introduction rather than other papers. However, the conceptual clarification applies equally to all active inference studies.

of Helmholtz's theory. Had they engaged with those ideas, they would have realized the 'problem of prior knowledge' and the difficulties with the above-mentioned move based on his authority.

Hatfield (2011) summarizes Helmholtz's three main theses that remained constant throughout his career:

- "Sensations are "symbols" or "signs" of their causes in objects. There is no similarity or resemblance between the character of the sensations and the properties in objects that cause the sensations."
- "All sensory nerve fibers produce sensations varying only in quality and intensity. These sensations lack spatiality, which develops through psychological processes of inference."
- "The law of cause guides these inferences."

The first thesis above is the basis of 'cutting off' an agent from its environment. According to Helmholtz, the first thesis implies a kind of skepticism about the 'external world' since we never have direct contact with them, but only with the activities of our sensory nerves, and these sensory nerves have 'no similarity or resemblance' with the 'external world.' This also includes the spatiotemporal properties of the 'external world,' for our sensations 'lack spatiality.'[4] To get the 'external world' from such ambiguous, impoverished, non-spatial sensations, Helmholtz posits mental/cognitive laws of association and inductive inferences. This is the second thesis above.

The crucial ingredients for carrying out these inferences and associations are what he called 'major premises.' As the third thesis above says, these inferences and major premises are grounded in the law of cause. Until this point, Helmholtz sticks to his empiricism and scientific impulse. However, at the last stage, he falls back on Kant's *transcendental category of causation* to ground his whole theory. In other words, it is through applying the a-priori, *transcendental* law of cause that unconscious inference gets going, which yields our perception. There are phases in his career where he even thinks space and time are not real but rather 'subjective and ideal' given that the starting data 'lack spatiality' (Hatfield, 1990). It is these transcendental categories that Helmholtz gives as answers to our 'problem of prior knowledge.'

It is such appeal to non-empirical, transcendental knowledge that we find as difficulties in Kant/Helmholtz's theory of perception as inference. Unless active inference theorists provide reasonable grounds for the origin and nature of the 'prior knowledge' other than nativism or transcendental categories, our challenge of inadequacy to them remains.

The main takeaway from our conceptual clarification in this section is that inferential perception, as an explanatory psychological/philosophical theory of perception, has a challenging problem. This problem is not a pressing issue in the engineering/robotics context of simulations owing to their addressability from the engineer's perspective. However, as a psychological theory of perception, it is inadequate and incomplete.

Obliviousness to the inadequacies of inferential perception leads to extravagant psychological/philosophical claims by active inference & Bayesian brain theorists like - 'Perception is an illusion,' 'Perception is controlled hallucination,' 'It is all hypothesis. It is all fantasy,' 'External world is our hypothetical best guess,' etc. Paying attention to these inadequacies might lead us to search for alternate ways to theorize perception and potentially model the causal relations involved in our perceptual experience in alternate mathematical terms. We will leave the mathematical modeling of such an alternate theory for the future but briefly describe its psychological and philosophical details in the next section.[5,6]

Direct Perception

Before we move into this section, we want to caution the reader of two things. As one might gather from the last few paragraphs of the previous section, we have slowly transitioned from mathematical psychology to purely theoretical & philosophical psychology. The mathematical part of the paper ends with the previous section, and the current section comprises the philosophical part, which constitutes our first caution.[7] The second caution refers to our reliance on the philosophy of Alfred North Whitehead to adumbrate a theory of direct perception.[8] We request the reader to take these two aspects of this section as axiomatic, for giving any more reasons/justifications would be difficult within the constraints of the paper.

The most important concept in Whitehead's philosophy that is immediately relevant for psychology is the concept of

[4] In some places, Helmholtz also talks about the two-dimensionality of these sensations rather than lacking any spatiality, but either way, there is no similarity or resemblance.

[5] An anonymous reviewer pointed out to us the relevance of Fodor (1984)'s theory of representation for this paper. Sticking only to the active inference part of their suggestion ('Active inference is firmly in the causal [theory of representation] camp'), we would like to object that it is not. For Fodor, a representation is something for which truth/semantic evaluation is appropriate. Specifically, for a causal theory of representation, R is a *true representation* of some state of affairs S if and only if R is *caused by* S - "'R *represents* S' *is true iff* C; C specifies some sort of *causal* relation between R and S" (Fodor, 1984). Our objection comes from the fact that in active inference there are not only causal conditions but unspecified a priori, transcendental conditions. In this sense, they are certainly not in Fodor's causal camp but rather have an *a priori* theory of representation.

[6] Another anonymous reviewer pointed to 'rigorous accounts of structure learning and prior formation' as potentially addressing the problems raised here. We fully agree with the reviewer but argue in Raghuveer and Endres (2023) that such an account is precisely what is missing in current active inference models (cf. Rutar, de Wolff, van Rooij, and Kwisthout (2022)).

[7] Our justification for the philosophical part comes from the fact that "[t]here can be no true physical [and psychological] science which looks first to mathematics for the provision of a conceptual model" (Whitehead, 1922).

[8] Our justification for this reliance has two reasons. One is the general mode of thought that Whitehead (along with William James and Henri Bergson) exemplifies - radical empiricism and process philosophy. The other is that Whitehead is the most systematic and mathematical of all radical empiricists and process thinkers.

'prehension' (and the related concept of 'objectification'). To understand the concept of prehension, one has to dive into its origin, which is Whitehead's theory of electrodynamics and gravitodynamics. For Whitehead, electrodynamics studies events (4-D spatiotemporal regions) characterized by the scientific object called 'electric charge,' while gravitodynamics studies events characterized by the scientific object called 'mass.' Like all events in nature, electrodynamic events also occur in the present and move into the past (for example, our perception of the external world occurs in the specious present and topples into the past).

Once an electrodynamic event perishes into the past (E_p), it transmits what is called a wave potential (or retarded wave potential) into its causal future and causally influences other electrodynamic events in its future. The same happens for gravitodynamic events (gravitational wave potential). The systematic causal influence on other electrodynamic events in the future is called the electromagnetic field streaming from E_p. Conversely, any electrodynamic event in the future (E_f) is causally influenced by *all* such electrodynamic events in its past. The systematic causal influence from *all* electrodynamic events in the past is the electromagnetic field streaming into E_f.

This systematic causal influence of past events on a future event can be given a precise mathematical description by means of the wave potentials. For example, in the case of gravitodynamics the mathematical description looks like: $dJ = \sqrt{dr^2 - \frac{2}{c^2}\sum_M \varphi_M dR^2}$ where φ_M is the wave potential associated with the gravitodynamic event of mass M. Whitehead gives similar description for electrodynamics and special relativity which was empirically indistinguishable from Einstein's special and general relativity (cf. Desmet (2010)).

The key element in the above equation is the summation symbol. Every electrodynamic and gravitodynamic event takes into account, 'appropriates,' and 'unifies' *all* (summation over masses M) the wave potentials from past events. This grasping-together and unification of potentials is what Whitehead calls a 'prehensive unification' or, simply, 'prehension.' Thus, every electrodynamic and gravitodynamic event is a complex prehensive unification of its past. The crucial point to note is that perished past events do not merely have external relations to the present but are internally related to the present. The causal past is internally related & immanent (as wave potentials) within the present.

Whitehead writes,

> "The various particular occasions of the past [E_p] are in existence, and are severally functioning as objects [wave potentials] for prehension in the present [E_f]. This individual objective existence of the actual occasions of the past, each functioning *in* [immanently, internally related to] each present occasion, constitutes the causal relationship which is efficient causation" (Whitehead, 1933, empahsis ours).

Thus, according to Whitehead, the theory of electrodynamics and gravitodynamics exhibits the empirical possibility of the immanence of the past *in* the present, the cause *in* the effect. Furthermore, from his understanding of the above theories, he asserts a generalized proposition that *any* event in nature is a prehensive unification of potentials of past events that are immanent and internally related.

Applying this generalization to psychology, we see that any perceptual event is a prehensive unification of all potentials of past events.[9] However, the potentials involved here are not just wave potentials (correlative to scientific objects) but sensory & perceptual potentials (correlative to sense objects and perceptual objects). Explaining Whitehead's theory of objects/theory of pure potentials is beyond the limitations of this paper. However, we have detailed it in our previous work (Raghuveer and Endres (2025)). It is this close analogy between an electrodynamic event and a perceptual event that leads to what we call a 'field-theoretic conception of perception.' Whitehead says, "The physicist's field of force and the psychological field both give you the same apprehension of "field.""

What is direct about this alternate conception of field-theoretic perception pertains to rejecting all three theses of Helmholtz (and, by extension, active inference). Firstly, there *is* a similarity or resemblance between scientific objects (electric charge, 'causes') and sense objects (green, 'sensations') in Whitehead's theory. They are both objects/adjectival characters of events. Secondly, immediate bodily events (if that is what Helmholtz means by 'sensory nerve fibers') & remote physical events are both events (4-D spatiotemporal regions), and *neither* of them 'lack spatiality.' Taken together, the above propositions imply that there is no need for unconscious mental inferences based on unaccounted transcendental knowledge of causation to get perception going. With this, we conclude our brief adumbration of the psychological & philosophical details of direct perception.

Conclusion

In conclusion, we repeat the paper's central thesis, which is that Active Inference, as it is currently formulated, is inadequate and incomplete in the domain of perceptual experience. Perception as inference faces the 'problem of prior knowledge,' which becomes evident when considered as a psychological theory of perception but gets masked/addressable in an engineering/technological context. We briefly described the psychological and philosophical principles involved in an alternate theory of perception, leaving its mathematical working out for future works.

[9] The justification for this generalization and application comes from the fact that, unlike the philosophies of materialism and dualism, Whitehead does not make an ontological distinction between physical experience (which is what physics studies) and perceptual/cognitive experience (which is what psychology studies). There is no bifurcation between electric charge and green ('causes' and 'sensations' in Helmholtz's terms). This is the essence of Whitehead's (and James' & Bergson's) radical empiricism.

Acknowledgments

This work was supported by the **DFG GRK-RTG 2271 'Breaking Expectations'** project number 290878970.

References

Badcock, P. B., Ramstead, M. J., Sheikhbahaee, Z., & Constant, A. (2022). Applying the free energy principle to complex adaptive systems. *Entropy*, *24*. doi: 10.3390/E24050689

Biehl, M., Guckelsberger, C., Salge, C., Smith, S. C., & Polani, D. (2018, 8). Expanding the active inference landscape: More intrinsic motivations in the perception-action loop. *Frontiers in Neurorobotics*, *12*, 387187. doi: 10.3389/FNBOT.2018.00045/BIBTEX

Clark, A. (2013). Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, *36*, 181-204. doi: 10.1017/S0140525X12000477

Desmet, R. (2010). *Whitehead's philosophy of mathematics and relativity*. Unpublished doctoral dissertation, Vrije Universiteit Brussel.

Fields, C., Fabrocini, F., Friston, K., Glazebrook, J. F., Hazan, H., Levin, M., & Marciano, A. (2023a, 6). Control flow in active inference systems - part i: Classical and quantum formulations of active inference. *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, *9*, 235-245. doi: 10.1109/TMBMC.2023.3272150

Fields, C., Fabrocini, F., Friston, K., Glazebrook, J. F., Hazan, H., Levin, M., & Marciano, A. (2023b, 6). Control flow in active inference systems - part ii: Tensor networks as general models of control flow. *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, *9*, 246-256. doi: 10.1109/TMBMC.2023.3272158

Fodor, J. A. (1984). Semantics, wisconsin style. *Synthese*, *59*(3), 231–50. doi: 10.1007/bf00869335

Friston, K. (2009, 7). The free-energy principle: a rough guide to the brain? *Trends in cognitive sciences*, *13*, 293-301. doi: 10.1016/J.TICS.2009.04.005

Friston, K. (2010, 1). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, *11*, 127-138. doi: 10.1038/nrn2787

Hatfield, G. (1990). *The natural and the normative: Theories of spatial perception from kant to helmholtz*. Cambridge: MIT Press.

Hatfield, G. (2011, 5). Kant and helmholtz on primary and secondary qualities. *Primary and Secondary Qualities: The Historical and Ongoing Debate*. doi: 10.1093/ACPROF:OSO/9780199556151.003.0013

Heins, C., Millidge, B., Demekas, D., Klein, B., Friston, K., Couzin, I., & Tschantz, A. (2022, 1). pymdp: A python library for active inference in discrete state spaces. *Journal of Open Source Software*, *7*, 4098. doi: 10.21105/joss.04098

Hohwy, J. (2014). *The predictive mind*. Oxford University Press. doi: 10.1093/ACPROF:OSO/9780199682737.001.0001

Millidge, B., Tschantz, A., & Buckley, C. L. (2021, 2). Whence the expected free energy? *Neural Computation*, *33*, 447-482. doi: 10.1162/NECO_A_01354

Oliver, G., Lanillos, P., & Cheng, G. (2022, 6). An empirical study of active inference on a humanoid robot. *IEEE Transactions on Cognitive and Developmental Systems*, *14*, 462-471. doi: 10.1109/TCDS.2021.3049907

Parr, T., Pezzulo, G., & Friston, K. J. (2022). *Active inference: The free energy principle in mind, brain, and behavior*. MIT Press.

Pezzulo, G., Rigoli, F., & Friston, K. (2015, 11). Active inference, homeostatic regulation and adaptive behavioural control. *Progress in Neurobiology*, *134*, 17-35. doi: 10.1016/J.PNEUROBIO.2015.09.001

Pezzulo, G., Rigoli, F., & Friston, K. J. (2018, 4). Hierarchical active inference: A theory of motivated control. *Trends in Cognitive Sciences*, *22*, 294-306. doi: 10.1016/J.TICS.2018.01.009

Raghuveer, D., & Endres, D. (2025). *Direct perception and whiteheadian natural philosophy.* (preprint on PhilArchive at `https://philpapers.org/rec/RAGDPA`)

Raghuveer, D., & Endres, D. M. (2023). Active inference and psychology of goals: A study in substance and process metaphysics. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *45*.

Ramstead, M. J., Badcock, P. B., & Friston, K. (2018, 3). Answering schrödinger's question: A free-energy formulation. *Physics of Life Reviews*, *24*, 1-16. doi: 10.1016/J.PLREV.2017.09.001

Rutar, D., de Wolff, E., van Rooij, I., & Kwisthout, J. (2022, 6). Structure learning in predictive processing needs revision. *Computational Brain and Behavior*, *5*, 234-243. doi: 10.1007/S42113-022-00131-8/FIGURES/3

Whitehead, A. N. (1922). *The principle of relativity: With applications to physical science*. Cambridge [Eng.]: The University press.

Whitehead, A. N. (1933). *Adventures of ideas*. Free Press.

Proceedings of the 23rd International Conference on Cognitive Modelling (ICCM 2025)

Adapting A Vector-Symbolic Memory for Lisp ACT-R

Meera Ray (mfr5832@psu.edu)
Department of Computer Science and Engineering, The Pennsylvania State University
W209 Westgate Building, University Park, PA 16802. USA

Christopher L. Dancy (cdancy@psu.edu)
Department of Industrial and Manufacturing Engineering, The Pennsylvania State University
310 Leonhard Building, University Park, PA 16802. USA

Abstract

Holographic Declarative Memory (HDM) is a vector-symbolic alternative to ACT-R's Declarative Memory (DM) system that can bring advantages such as scalability and architecturally defined similarity between DM chunks. We adapted HDM to work with the most comprehensive and widely-used implementation of ACT-R (Lisp ACT-R) so extant ACT-R models designed with DM can be run with HDM without major changes. With this adaptation of HDM, we have developed vector-based versions of common ACT-R functions, set up a text processing pipeline to add the contents of large documents to ACT-R memory, and most significantly created a useful and novel mechanism to retrieve an entire chunk of memory based on a request using only vector representations of tokens. Preliminary results indicate that we can maintain vector-symbolic advantages of HDM (e.g., chunk recall without storing the actual chunk and other advantages with scaling) while also extending it so that previous ACT-R models may work with the system with little (or potentially no) modifications within the actual procedural and declarative memory portions of a model. As a part of iterative improvement of this newly translated holographic declarative memory module, we will continue to explore better time-context representations for vectors to improve the module's ability to reconstruct chunks during recall. To more fully test this translated HDM module, we also plan to develop decision-making models that use instance-based learning (IBL) theory, which is a useful application of HDM given the advantages of the system.

Keywords: ACT-R; vector-symbolic architectures; Lisp; distributional semantics

Introduction

In this paper, we describe our ongoing efforts to adapt HDM to work better with Lisp-based ACT-R, thus bridging a newer vector-symbolic model with decades of cognitive modeling done with Lisp ACT-R. Our eventual goal is to build a cognitive agent that simulates the actions taken by disaster survivors and the social factors that impact those decisions. We chose ACT-R due to the need for the cognitive agent to represent the cognitive processes of a disaster survivor well enough to accurately model the decisions they would make, relative to the empirical results gathered from human subjects; ACT-R has successfully modeled the decision making of humans in constrained experiments (Ritter, Tehranchi, & Oury, 2019), sometimes serving as a useful lower-level cognitive representation for rational level (i.e., Newell, 1990) decision-making theories (e.g., Gonzalez, Lerch, & Lebiere, 2003). At the same time, the model needs to include situational awareness to accurately model how different survivors will respond to the same disaster environment differently depending on their social background (Prather et al., 2022). Here we use situational awareness to mean background knowledge, context, and worldview that puts the perceived environment into context (Lukosch, Bekebrede, Kurapati, & Lukosch, 2018). We hypothesize that a vector-symbolic memory architecture, Holographic Declarative Memory (Kelly, Arora, West, & Reitter, 2020), can be useful for representing this context by reading in related texts into memory, which in turn influences which chunks are retrieved at runtime and which procedures are run. Furthermore, the increased scalability of a system like HDM may be more appropriate for representing the impact of existing social structures, which may themselves be more latently represented in memory in a way that requires a large number of concepts and associations to adequately represent.

In the next sections, we first justify why it is worthwhile to use HDM when ACT-R already has a built-in Declarative Memory system. Next, we explain adaptations needed at the architectural level for (canonical, Lisp) ACT-R and HDM for integration. We conclude with a discussion of preliminary results of testing this new translated HDM and future work.

What Does HDM add to ACT-R?

Traditional ACT-R models have two main memory components for modelers to manipulate: procedural memory, typically production rules written by the developer, and declarative memory, which may store semantic or episodic information that is often entered by hand before the model runs (Ritter et al., 2019). ACT-R retrieves relevant memories based on environmental cues specified as requests in production rules. However, by default ACT-R's declarative memory (DM) system lacks partial memory recall, meaning a gradation of which chunks are relevant to a query, as opposed to returning either one matching chunk or none at all. Additionally, the time complexity to add and retrieve chunks as well as the space complexity to store chunks would be prohibitive to adding corpus-sized amounts of text to DM (Kelly et al., 2020). Holographic declarative memory (HDM) addresses these problems by representing each unique input token in vector space rather than storing the entire input, which allows for a continuous similarity measure between queries and memories. HDM can thus "read" large corpora of relevant text into a model's memory; it is thus a distributed semantics model like Word2vec made available as a memory system for

a cognitive architecture. Despite the machine-learning-like principle behind it, HDM still achieves high-level cognitive plausibility in free recall effects and significantly fits human performance in a decision task.

Thus, HDM presents a useful alternative to the default ACT-R declarative memory representation, especially for models that may need to represent sociocultural structures in memory and represent similarity between concepts or chunks in an architecturally-defined way. The latter ability is helpful for implementing theories in ACT-R that rely on context-dependent associations and similarities, such as Instance-Based Learning (IBL) Theory (Gonzalez et al., 2003). This vector-based representation may also make architecture-level integration with generative models (which typically also use vector-based representations for text-based "tokens") more straightforward (e.g., see Dancy & Workman, 2023).

Fortunately, the implementations of both HDM and the newest (Lisp) ACT-R provide enough of a foundation for a connection between the two systems. Nonetheless, to integrate HDM into the canonical version of ACT-R, both systems need to be adapted. This need for adaptation and expansion of HDM is especially true if one wants to use HDM while keeping declarative memory use patterns that are common amongst ACT-R modelers.

Adapting ACT-R for HDM

We used the interface provided in the canonical ACT-R 7 Python connection files (ACT-R Research Group, 2023) and the HDM code Dr. Kelly wrote for Python-ACTR (Kelly, 2020) to implement the Lisp-side ACT-R commands with Python code. At present, we have implemented the core commands for adding to and retrieving from memory as well as convenience and utility commands like `(sgp)` and `(dm)`. The command `(dm)` now prints the unique values stored by HDM rather than chunks, which are not explicitly stored, as well as a two-dimensional visualization of the HDM vector space. Goal chunks, which require more precise recall and typically do not come in a large scale number of chunks, are to be created with `(define-chunk)` rather than `(add-dm)` so ACT-R's default systems are used rather than HDM. The parameters for the whole-chunk recall mechanism, which will be explored in the next section, can be set globally like any other parameter in `(sgp)` or locally in any memory retrieval request.

Adding Text to HDM in ACT-R

We added the ACT-R command `(preprocess-text)`, which takes a raw plain text file and outputs a file ready to be added to memory. The preprocessing step uses the Natural Language Toolkit (NLTK) to remove *stopwords*, often-appearing words such as "and" or "the," which don't convey information specific to a task (Sarica & Luo, 2021).

We made the decision to read the text into HDM sentence by sentence, though (Kelly et al., 2020) don't indicate whether or not this is necessary. When HDM encodes a list of values, it stores associations between all pairs of words

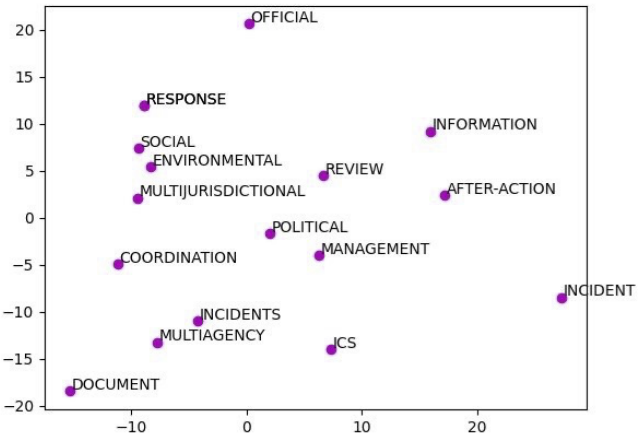


Figure 1: PCA Plot of HDM Vectors

in left-to-right order. For words that are far apart from each other, it is less worth computational resources to encode their associations with each other. Other ways of splitting the text may be optimal, but we chose the sentence as the unit to add at a time. Therefore, the preprocessing step tokenizes the text into sentences using NLTK's Punkt pretrained tokenizer to find sentence boundaries.

From the preprocessed file, the modeler uses the `(read-corpus-hdm)` command to read in the preprocessed file to HDM. As implemented, this step can only take place after the model has been loaded. An example of a plot of HDM vectors after being added to memory is Figure 1. The corpus here is from a document instructing U.S. officials on how disaster response and recovery are organized.

Adapting HDM for ACT-R

A downside of HDM is that, unlike ACT-R DM, it lacks a mechanism to retrieve a chunk, a group of memories bound together, in its entirety without knowing the slot corresponding to each unknown value being requested. For example, a chunk "homeowner:yes damage:severe renter:no neighborhood:Eastville" could represent the status of an disaster survivor agent's home after a flood; note the format of slot:value pairs. A cue to retrieve the chunk in DM would simply include a unique slot:value pair of the chunk — "homeowner:yes". With HDM, one would have to specify that pair and then ask for one unknown at a time — "homeowner:yes damage:?", "homeowner:yes renter:?" and so on. We extended the HDM implementation using a simple chaining method so multiple unknowns can be requested, but every slot in the chunk still needs to be specified; this is cumbersome for a modeller compared to ACT-R's original DM. For us, it would require us to write more and longer production rules for our cognitive agent. Thus, we devised a method of full-chunk retrieval without storing chunks outright or deviating from HDM's memory vector approach.

We define the problem as follows: after adding chunk

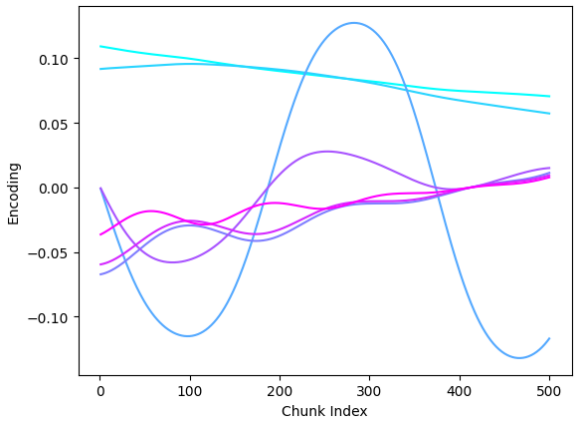


Figure 2: Oscillator function vector $\mathbf{T}(t)$ as a function of chunk index t. The functions plotted above are $T_1(t), T_{51}(t), T_{101}(t), T_{151}(t), T_{201}(t), T_{251}(t), T_{301}(t)$ out of the $\mathbf{T}(t) = [T_1(t), T_2(t), ...T_{320}(t)]$. Note how the oscillators vary from lower to higher frequencies. While some individual functions repeat in encoding value over the chunk index range, the vector at any one chunk index is unique.

$\mathbf{c} = \{s_1{:}v_1, s_2{:}v_2, ...s_n{:}v_n\}$ into memory, retrieve the entire chunk with a cue $\mathbf{q} = \{s'_1{:}v'_1, s'_2{:}v'_2, ...s'_m{:}v'_m\}$, where $\mathbf{q} \subseteq \mathbf{c}$ and the slots and their associated values are $s_1..s_n$ and $v_1...v_n$ respectively.

Time Encodings

First, we needed a way to represent which bits of memory were entered together as a chunk at the same time. We turned to neural oscillator encodings of serial memory (Brown, Preece, & Hulme, 2000) for a biologically-plausible theory of temporal encoding. In their model, there are fifteen oscillator functions $O_1(t)...O_{15}(t)$ of timestep t which range from lower to higher frequency, with R and ϕ as sources of noise:

$$O_i = sin(\phi + t\theta_i) \quad (1)$$

$$\theta_i = SR2^i \quad (2)$$

$$R \sim \mathcal{N}(0, \sigma^2) \quad (3)$$

$$\phi \sim \mathcal{U}(0, \pi/\theta) \quad (4)$$

S is a scaling parameter that can be swapped out for different time scales. Brown et al. (2000) sets $S = 10^{-5}$, but we've found different values work better on different time scales (e.g. number of chunks). For the variance σ^2 of R, $\sigma^2 = 1$ in the Brown model but can be increased or decreased to increase or decrease the amount of noise.

The purpose of having noisy representations rather than simply, for example, sequentially numbering each chunk index, is to model the recall of human subjects in serial order memory tasks. An incorrect item is more likely to be exchanged for a correct item during recall the closer the two items are in position. When positions are hierarchical, e.g. lists of lists, often the position of an item within a list is recalled correctly even if the list's position compared to other lists is not. For example, one may recall an item being in the third value of the second chunk instead of the third value in the first chunk. Multiple oscillating functions represent this hierarchical error by creating "bumps" in similarity between cognitive representations of two items even as their similarities decrease overall as their positions grow further apart.

To work more simply within the HDM framework, we flattened Brown et al. (2000)'s sixteen "learning context" vectors of twenty elements each into a single 320-element time vector $\mathbf{T}$ for each timestep. Each element T_i of the time vector in Equation 5 randomly draws four of the 15 oscillators O_j from Equation 1, with cosine being randomly substituted for sine with a probability of half:

$$T_i = \prod_{j=1}^{4} \sin(O_j) \quad (5)$$

The function $\mathbf{T}(t) = [T_1(t)...T_{320}(t)]$ is generated and stored when the model is loaded. For each chunk added to memory, the time step t increases by 1 and the time vector $\mathbf{T}(t)$ is calculated. Note that the four oscillator functions randomly chosen above are not chosen with uniform probability; they're selected so the low-frequency oscillators make up a larger share than the higher-frequency oscillators. We did this because we found too many high-frequency oscillators led to nearly-repeating vectors over a small period of time. The probability distribution we selected was a discretized and bounded modification to the exponential distribution. We found it to be the best match for how the oscillators are distributed in Figure 6 of Brown et al. (2000). The best match for the figure is with a scaling parameter of $\beta = 5.125$. However, the distribution can be used with different scaling parameters for learning context vectors of arbitrarily chosen sizes if one adjusts the scaling parameter to be the expected value of which oscillators is chosen. For example, the scaling parameter β being 5.125 means the center of the probability mass is between the sixth and seventh oscillator (zero indexing). A plot of an oscillator vector is shown in Figure 2. We used the default parameters described earlier ($\sigma^2 = 1, \beta = 5.125$) but set the time scaling parameter $R = 5 * 10^{-6}$ to reflect the large range of chunk indices from 1 to 500.

Time HDM Vectors

Now that we have a continuous time representation, we need to somehow bind the representation with HDM's memory vectors. HDM itself uses the holographic reduced representations (HRR) framework (Plate, 1995). Plate shows how memory traces can be constructed by binding together two "holographic" vectors $A, B \in \mathbb{R}^n$ using circular convolution ($\circledast$). From a memory trace $C = A \circledast B + ...$ and a cue A, a noisy version of B can be recovered. Simply using the time vector itself as a holographic vector would violate HRR's assumption that bound vectors are drawn from the same distribution.

HDM's holographic vectors are simply drawn from the normal distribution, while the time vectors are oscillating outputs of $\mathbf{T}(t)$. Instead, we needed to treat the time vector as a kind of value being stored. The problem is that HDM, like ACT-R, works with discrete memory values even if it encodes them with continuous values. To encode the continuous time vector, we adopted the fractional binding operation from Komer, Stewart, Voelker, and Eliasmith (2019) and composed it with $\circledast$ from Plate to arrive at a holographic representation $\widetilde{\mathbf{T}}$ of the time vector $\mathbf{T}$.

$$\widetilde{\mathbf{T}} = \mathcal{F}^{-1}\left\{\sum_{l=1}^{320} \mathcal{F}(e_{t_l})^{\mathbf{T}(t)}\right\} \tag{6}$$

The function $\mathcal{F}$ above is the Fourier transform. The HRR vectors $e_{t_1}...e_{t_{320}}$ are generated from the same distribution as HDM's environment holographic vectors. Next, we associate the chunk's time vector with each bit of memory in the chunk. We store these associations in *mt* vectors – Each HDM memory vector m has an associated time-memory vector mt. This does double the memory usage but keeps the same space complexity class. When a chunk $s_1 : v_1, ...s_n : v_n$ is added to memory, a $\widetilde{\mathbf{T}}$ is created for it as per Equation 6 . The *mt*s are updated with the following rule:

$$mt \longleftarrow mt + \widetilde{\mathbf{T}} \circledast \sum_{c=1}^{n} s_c \circledast v_c \tag{7}$$

where s_c and v_c are corresponding slot-value pairs in a chunk.

This update rule, unlike the update rule for HDM's memory vectors, does not add noise. From an implementation standpoint, adding noise would interfere with the recalling of added traces in Plate (1995), which already has some noise in reconstruction. From a cognitive processes representation standpoint, the noise to represent cognitively plausible time-memory errors is already represented in the time vectors themselves as discussed earlier.

To retrieve the chunk $\mathbf{c}$ from cue $\mathbf{q} = \{s'_1{:}v'_1, s'_2{:}v'_2, ...s'_m{:}v'_m\}$, we construct HRR Q from $\mathbf{q}$, take its inverse, and bind it with each memory-time vector *mt* to obtain a noisy reconstruction $\hat{\mathbf{T}}$ of the time HRR:

$$Q = \sum_{c=1}^{m} s'_c \circledast v'_c \tag{8}$$

$$\hat{\mathbf{T}} = mt \circledast Q^{-1} \tag{9}$$

The retrieval works because $\circledast$ is distributive, so *mt* contains $\widetilde{\mathbf{T}} \circledast s_c \circledast v_c$ for each slot:value pair. Note that the inverse operation used here is the approximate inverse from Section VII, Subsection C in (Plate, 1995).

We are left with $\hat{\mathbf{T}}_1...\hat{\mathbf{T}}_N$, the collection of reconstructed time vectors for each of the N *mt*s. As per Brown et al. (2000), we step through the time vector function $\mathbf{T}(t)$ from $t = 1$ to the current time step in the model, comparing at each time step which of the reconstructed time vectors $\hat{\mathbf{T}}$ have the greatest similarity with the current $\mathbf{T}(t)$. Depending on the method chosen — taking the top p results, all results above a threshold, or results meeting the previous two criteria — we end up with a list of possible slots and values making up a chunk. HDM tracks which memory vectors are slots, so we keep the *mt*s corresponding to slots and then build a complete cue $\mathbf{q} = \{s_1 :?...s_n :?\}$ that contains every slot. The chunk can then be requested using multiple unknown request chaining, mentioned at the beginning of this section, with HDM's built-in retrieval mechanism. Thus, we are in principle able to retrieve the entire chunk.

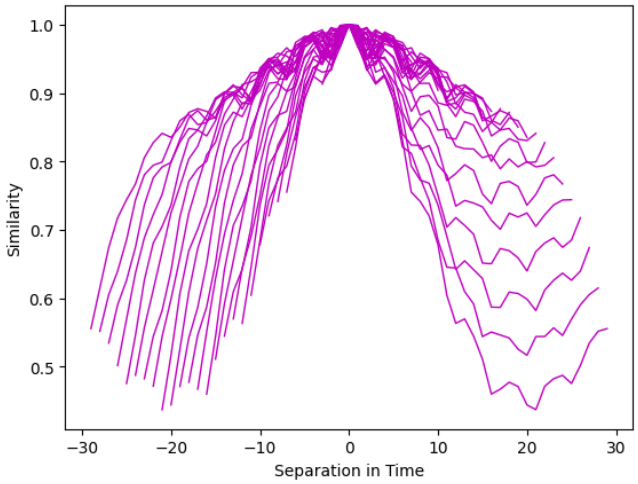


Figure 3: The dot product similarity between our time vector $\mathbf{T}(t') \cdot \mathbf{T}(t'')$ (y axis) and $t' - t''$ on the x axis. It peaks at 1 (comparing identical vectors) and has smaller local maxima to represent noise in serial memory recall.

It's worth noting that, like HDM, the time complexity scales with the number of unique values rather than the total number of chunks. Still, an option to only turn on slot pulling encoding for chunks of interest so only a subset of relevant time-memory vectors are stored would reduce the retrieval times. However, remembering chunks (e.g. sentences) stored during the corpus adding step could be useful for some modeller applications as well.

Preliminary Results and Discussion

Time Encodings

We generated two time vector functions using exactly the same set of parameters: $\sigma^2 = 1$ for Equation 3, $S = 10^{-5}$ for Equation 2, and $\beta = 5.125$ as the exponential distribution scaling parameter to sample four oscillators from the 15 oscillator functions to generate each element in Equation 5. The time vector has 320 elements as per our adaptation of Brown et al. (2000).

In order to evaluate the quality of the time vector representations, we plotted a similarity function $\mathbf{T}(t') \cdot \mathbf{T}(t'')$ for all $t', t'' \in \{1, 2, ..30\}$. The similarity function $\cdot$ is the dot product, which is equivalent to cosine similarity here since

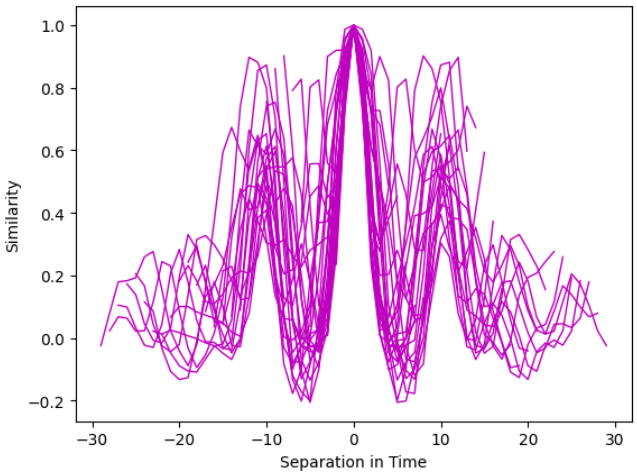


Figure 4: A self-similarity graph like Fig. 3 but with much larger "bumps," indicating a noisier representation of serial memory.

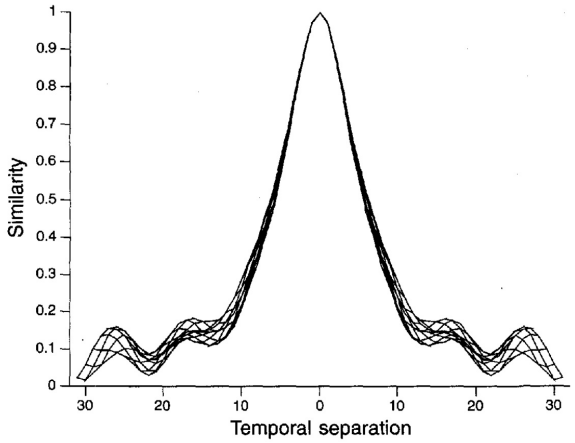


Figure 5: The (averaged) self-similarity of time vectors in Brown et al. (2000). Plot reproduced from Figure 7B of (Brown et al., 2000)

the vectors are normalized. The similarity plots are shown in Figure 3 and Figure 4.

As noted earlier, a similarity function between time vectors taken at subsequent time steps should show two properties: (1) a gradual decrease as the times grow further apart and (2) some oscillation to represent the cognitive error that arises from time's hierarchical representations. Our time encodings do show both of these properties, but the "bumps" that represent the error are more pronounced than in self-similarity plot of Brown et al. (2000), reproduced here in Figure 5. Both plots are over the same time step range and equal corresponding parameters as our plot. A possible explanation is that the dot product between two large vectors rather than the average dot product between two sets of smaller vectors results in more noise, which Brown et al. (2000) mentioned as a justification for his choice of the latter representation.

Furthermore, one can observe a stark difference between the self-similarity of Figure 3 and Figure 4 despite having oscillator parameters generated from the same probability distributions with the same parameters. In our tests, neither increasing nor decreasing σ^2 had much consistent effect on how much variability appeared between the self-similarity plots of time vector functions. Increasing or decreasing the exponential distribution scale parameter β to respectively favor or disfavor slower oscillators had surprisingly little effect either on the variance between time functions or on the level of noise any particular time function showed in self-similarity plots. Further study is needed to quantify the self-similarity noise in a single comparable metric and decide which parameters yield representations that best fit observed cognitive effects in serial order.

Conclusion

So far, we have described how full-chunk recall can be supported without needing to store the chunks themselves nor a data structure scaling in size with the number of associations between values in the chunk. The time vector representations we have implemented so far are noisy but still provide distinct representations for distinct chunks. Work to measure slot pulling effectiveness across a wider variety of models is ongoing.

In the future, we want to try another time encoding for associating chunks together besides neural oscillators. We also want to examine different chunk retrieval mechanisms. So far, we take the two inputs —reconstructed time vectors per time-memory vector and the time vectors themselves — and compute the similarity between them as the only ranking criterion. However, we should also expect that the time vector reconstructions associated with the same chunk ought to cluster together in vector space. Perhaps the true time vectors represent cluster centers and a clustering measure could aid accurate and precise retrieval.

The vector-symbolic approach used here also presents an opportunity for more straightforward considerations of ways to integrate generative models within ACT-R. Such an inte-

gration would hold implications for expanded sociocultural representations in a cognitive architecture like ACT-R, as argued by Dancy and Workman (2023). As previously mentioned, having these sociocultural representations in memory would also give an opportunity to explore the impacts of sociocultural structures on decision making behavior given the interdependence between memory and decisions. We plan to explore interactions between sociocultural structures, memory, and decision making using HDM (and eventually generative models) in the context of IBL.

Adequately representing these sociocultural structures in memory requires memory representations to be scaled up efficiently. At the same time, there is value in making production systems work reliably with vector memory; production rules give explainable reasoning for decision making that generative models often struggle with. Full-chunk recall is needed to support flexible, reliable production systems interacting with vector-based memory. Also important is that full-chunk recall makes sure memories specific to an agent rather than its broader worldview are used at the appropriate times during decision making. The full-chunk recall mechanism and text-to-memory pipeline explained here hold promise for computational cognitive architectures like ACT-R to model the cognitive role that social structures play in human decision making.

References

ACT-R Research Group. (2023, July). *ACT-R ACT-R 7.27.9 software.* Retrieved 2023-12-06, from `https://act-r.psy.cmu.edu/act-r-7-27-9-software/`

Brown, G. D. A., Preece, T., & Hulme, C. (2000). Oscillator-Based Memory for Serial Order. *Psychological Review*, *107*(1), 127–181.

Dancy, C. L., & Workman, D. (2023). On integrating generative models into cognitive architectures for improved computational sociocultural representations [Conference Proceedings]. In *Aaai fall symposium series* (Vol. 2, p. 256-261). Washington, DC: AAAI Press.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making [Journal Article]. *Cognitive Science*, *27*(4), 591-635. doi: 10.1207/s15516709cog2704_2

Kelly, M. A. (2020, July). *ecphory/HDM.* Retrieved 2023-09-25, from `https://github.com/ecphory/HDM` (original-date: 2020-03-29T22:27:50Z)

Kelly, M. A., Arora, N., West, R. L., & Reitter, D. (2020). Holographic Declarative Memory: Distributional Semantics as the Architecture of Memory. *Cognitive Science*, *44*(11), e12904. doi: 10.1111/cogs.12904

Komer, B., Stewart, T. C., Voelker, A. R., & Eliasmith, C. (2019, July). A neural representation of continuous space using fractional binding. Montreal, Canada.

Lukosch, H. K., Bekebrede, G., Kurapati, S., & Lukosch, S. G. (2018, June). A Scientific Foundation of Simulation Games for the Analysis and Design of Complex Systems. *Simulation & Gaming*, *49*(3), 279–314. doi: 10.1177/1046878118768858

Newell, A. (1990). *Unified theories of cognition* [Book]. Cambridge, MA, USA: Harvard University Press.

Plate, T. (1995, May). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641. doi: 10.1109/72.377968

Prather, R. W., Benitez, V. L., Brooks, L. K., Dancy, C. L., Dilworth-Bart, J., Dutra, N. B., ... Thomas, A. K. (2022). What Can Cognitive Science Do for People? *Cognitive Science*, *46*(6), e13167. doi: 10.1111/cogs.13167

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2019). ACT-R: A cognitive architecture for modeling cognition. *WIREs Cognitive Science*, *10*(3), e1488. doi: 10.1002/wcs.1488

Sarica, S., & Luo, J. (2021, August). Stopwords in technical language processing. *PLOS ONE*, *16*(8), e0254937. (Publisher: Public Library of Science) doi: 10.1371/journal.pone.0254937

Simulation of the Affect of Emotions on Social Adaptation through Risk Adjustment of Intention Estimation

Ruiki Kawaji (kawaji.ruiki.16@shizuoka.ac.jp),
Junya Morita (j-morita@inf.shizuoka.ac.jp)
Department of Informatics, Graduate School of Integrated Science and Technology,
Shizuoka University, 3-5-1, Johoku, Hamamatsu Chuo-ku, Shizuoka, 432-8011, Japan

Hirotaka Osawa (osawa@a3.keio.jp)
Department of Industrial and Systems Engineering, Faculty of Science and Technology, Keio University,
3-14-1, Hiyoshi, Yokohama Kohoku-ku, Hamamatsu, Kanagawa, 223-8522, Japan

Abstract

This study was conducted to clarify how emotions contribute to social adaptation. Focusing on intention estimation and risk judgment as factors, a simulation was conducted using the cooperative game "Hanabi." The model in this study estimates intentions by recalling past instances of cooperation. A model was constructed in which emotions fluctuate depending on the outcome of cooperation and instance recall, and the results showed that emotions are useful in models corresponding to novices of the game, but their influence is reduced in models corresponding to experts. In addition, the results showed that cooperation failure due to risk acceptance decreased as emotions fluctuated, suggesting that emotions are an ability to minimize the risk of failure in intention estimation.

Keywords: ACT-R, Cognitive modeling, Cooperative behavior, Emotion, Hanabi

Introduction

Human social behavior is greatly influenced by emotions, which is considered as an evolutionary product to promote cooperation and socialization among humans, and has contributed to the survival of humankind (Cosmides & Tooby, 1997). Social behavior is based on the process of inferring intentions of others from their observable actions. It is said that the ability to model other's internal belief is necessary for intention estimation, and in the field of cognitive science, this ability is expressed as the term "theory of mind" (Premack & Woodruff, 1978). Both emotions and intention estimation are human abilities related to social behavior, and if the presence of emotions leads to socialization, then it can be thought that they support intention inference, which is an ability necessary for social behavior.

However, intention estimation is not always successful: it involves risk due to uncertainty caused by individual differences in perception of the environment. The tendency to accept or avoid risk is thought to be related to emotions (Turner, Zangeneh, & Littman-Sharp, 2006; Yuen & Lee, 2003). This relationship seems to be caused by the influence of the past experience on the current emotions (Bower, 1981). Past experience is crucial for estimating intention (Morita et al., 2018), as emotions influence behavior through associated memories. In this way, emotions influence on human behavior, but this influence is thought to differ depending on the situation. Theories regarding relationship between task difficulty and effect of emotions have been proposed for a long time, such as the flow theory (Czikszentmihalyi, 1990) and the optimal arousal level theory (Yerkes & Dodson, 1908). In addition, the influence of emotions also changes with expertise. Recent research into cognitive models has proposed a theory in which the influence of emotions embedded in experiences is folded into the process, reducing the impact of emotions on cognitive tasks (Conway-Smith & West, 2024).

In this study, we conduct a simulation of a cooperative game using a cognitive model to observe the influence of emotions on intention estimation in the above-mentioned situation. The simulation was conducted based on the hypothesis that emotions are useful in memory-based intention estimation by leading proper risk management It is also thought that such an effect of emotions in intention estimation is influenced by expertise in the task and task difficulty. From the difference in behavior between the models manipulating the above factors, we will discuss the role of emotions as an ability to smoothly carry out social behavior.

The Rules of Hanabi

In this study, we focus on Hanabi, a game that requires cooperation between players through estimating each other's intentions. In the past, research has been conducted on cooperation and intention estimation in this game (Osawa, 2015; Miyata & Osawa, 2024), and computer models (Kuwabara et al., 2023; Kawaji, Morita, & Osawa, 2024) has been constructed.

Hanabi is a game for two to five players. For simplicity, this study only deals with 2-player play. The game includes 50 cards in five colors (white, red, blue, yellow, and green) (three cards of each color with the number 1, two cards from 2 to 4, and one card with the number 5) and two types of tokens, red and blue. The goal of the game is to work together to stack cards of the same color in ascending numerical order and put as many cards on the table as possible.

At the start of the game, the cards are shuffled and five cards are dealt to each player, which become the player's hand. Players cannot see the contents of their own hand. Only other players can see the contents. Cards that are not in the hand become the deck. The first player's turn begins with

eight blue tokens and zero red tokens shared by all players.

Players are given their turn in order, and they must take one of the following three actions before moving on to the next player's turn.

- Hint: One color or one number from the cards in partner possession is selected and all the cards with that color or number are told to the partner. When the hint is given, one blue token is consumed. If there are no blue tokens and cannot pay the cost, this action cannot be performed.
- Discard: One card from the player's hand is discarded and one new card from the deck is added to the player's hand. The discarded card can be seen by everyone, including the player oneself, and cannot be used during the game. When the player discards a card, one blue token is restored. This does not allow the number of blue tokens to exceed the initial value.
- Play: A card from the self hand is selected and revealed. If the value of the played card is exactly one greater than the highest value of the card of the same color that has been already played, it is judged to be successful and the card is placed on top of the card of the same color. If there is no card of that color placed, it is treated as a number 0 card and the player can place a number 1 card. Otherwise, it is judged as a failure and the played card is discard. In the case of failure, the blue tokens are not restored and the players gain one red token. After that, whether the player succeeds or fails, the player adds one card from the deck to the player's hand.

There are three conditions to end the game. The first is to gain three red tokens. The second is for all players to act once after no cards are left in the deck. The third is for all five colors of fireworks to be stacked and completed. If the end conditions are reached, the score is calculated as the total number of cards stacked on the field, with a maximum of 25 points for 5 colors $\times 5$ cards. Scores can only be obtained by playing, but there is a high risk that the game will end if players fail three times. They need to give each other hints, but because they are only given a limited amount of incomplete information, they need to cooperate to figure out each other's intentions.

Model

The model in this study was developed based on Kawaji et al. (2024). This model uses the cognitive architecture ACT-R (Anderson, 2007) to perform intention estimation based on the use of instances via parameters corresponding to emotions. Furthermore, to model the expertise in collaboration, we implemented production rules for deciding on actions based on heuristics. Figure 1 is a flowchart showing the actions of the model from the start to the end of a turn.

Decision heuristics

The model assigns priorities to production rules to determine an action (Miyata & Osawa, 2024). There are six production rules, each with the following priority:

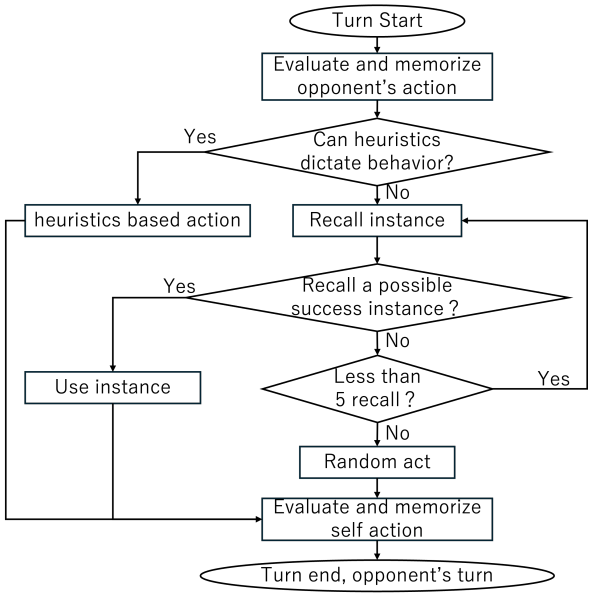


Figure 1: Flowchart of the model

1. Play a confirmed card: If the model can successfully play a card in the model's hand with a known color and number, play that card.
2. Hint for playable cards: In a situation where hints can be given, if the partner has a card that will be successfully played and that is given a hint about only one of the color and number, give a hint about the information of the card that is not yet known.
3. Hint for single cards: In a situation where hints can be given, if the model has a card that will be successfully played and there is no card of the same color or number in their hand, give a hint about the single card.
4. Discard a confirmed card: If the model has a card in the model's hand whose color and number are both known and the card of the same color and number is already successfully played, discard that. Or, if the model has two or more cards of the same color and number and both are confirmed, discard that.
5. Hint for playable cards: In a situation where hints can be given, if the partner has a card that will be successfully played, give a hint on the information of the card that is not yet known.
6. Hint for discardable cards: In a situation where hints can be given, if the partner has a card that has the same color and number as already been played successfully, give a hint on the information of the card that is not yet known.

Instance-based decision making

If none of the above heuristics can be applied, an attempt is made to determine an action based on instances, which are obtained after oneself or the partner performs the action. The information contained in the instances consists of the following:

- Type of the action: It classifies the instance into the three actions (hint, discard, or play).
- Outcome of the action: It indicates the action was successful or not. A successful play and a successful hint show an instance of self play leading to addition of points and an instance of hints leading to successful plays by partner, respectively. In the case of a discard, an instance of discarding card with the same color and number as the successful card that has already been played is judged as successful. All other instances are coded as failure.
- Target information: It indicates the color and number of the target card. If the action is performed without knowing the color or number, it is stored as unknown.
- Cards on the field: It indicates up to which numbers of each of the five colors have been played.
- Observable cards by the model: It indicates information on the model's hand that is known from the hint.
- Observable partner's cards by the partner: It indicates information on the hint the model gave them.
- Complete partner's card information: It indicates complete information on the partner's hand, including cards that the partner cannot see.
- Remaining cards: This is the number of cards that have not yet been viewed, calculated by subtracting the number of viewed cards from the total number of cards of each color and number.
- Last action: It clasifies the previous action that the model or the partner took most recently into three type of action (play, hint, discard).

The stored instances are used in the procedure shown in Figure 1. First, past instances are searched for based on the current situation. In this case, the ACT-R partial matching is used to recall the most similar instance.

If an instance is searched for and a successful instance is found, an attempt is made to act according to that instance. If the recalled instance is unsuccessful, if nothing can be recalled, or if an instance is recalled but the action cannot be performed according to the instance, recall is made again from instances other than those already recalled in this decision. Recall is repeated up to five times. With each repetition, the mismatch penalty is halved to increase randomness. If no instance is found after five repetitions, an action is decided randomly.

In the partial matching mechanism of recall, the activation value A_i of memory i is calculated by the following equation:

$$A_i = B_i + \sum_l PM_{li} + \varepsilon \tag{1}$$

B_i is the base level calculated based on the frequency of memory usage and the time since it was used, ε is noise, and P is the mismatch penalty coefficient (mp). M_{li} is the degree of match between each element l in instance i and the current situation.

For information about the model and the partner's hand, the cosine similarity (minus one) of the vector whose elements are the frequency of each color and number is used. "Type of the action", "target information", and "last action" take binary values (0 or -1) of the similarity according to whether they match or not. Regarding "outcome of the action," the similarity is calculated according to the influence of emotions shown in the next section.

These similarities are added up and the most similar instance is recalled. When an instance is recalled and the action has been succeeded, the same action as the instance is performed.

Emotional influence

The model's emotions are represented as two parameters, valence and arousal, based on the circumplex model (Russell, 1980). These parameters have a minimum of -1 and a maximum of 1, and the recall of instances changes depending on their values. To determine specific values, we constructed two models: fixed emotion and emotion fluctuation.

Fixed emotion In the fixed emotion model, the emotional parameters affect instance recall as a calculation of similarity. The current valence is V, and the outcome of the target instance is V_i; the difference between them is calculated as follows:

$$M_{li} = -|V - V_i|/2 \tag{2}$$

In this equation, V_i will be 1 if it is a success instance, as it is completely positive, and -1 if it is a failure instance, as it is completely negative. This is a cognitive model expression of the mood congruence effect (Bower, 1981) as the influence of emotional valence on recall. By labeling a memorized instance as positive (success), or negative (failure), the distance from the current emotional valence can be measured. Such coding makes recalling similar emotional experiences easier.

Arousal also affects B_i in equation 1. The offset value of B_i parameter is changed by arousal from 0.5 to 1.5. This makes it easier for recall to succeed when arousal is high, and easier for it to fail when arousal is low.

Emotion fluctuation In the emotion fluctuation model, the two parameters are updated based on the idea of prediction error: event matching predictions moves pleasantness, while moving it away evokes unpleasantness. Emotional dynamics differ by decision strategy:

- Heuristic actions rarely change emotion, because they almost always succeed and the gap between prediction and outcome is minimal.
- Random actions likewise cause small fluctuations; both prediction and outcome are imprecise, so the error remains ambiguous.
- Instance-based actions trigger large swings: success sharply reduces the error, producing strong pleasantness, whereas failure enlarges it, producing strong unpleasantness.

Table 1: Act and Emotional Changes

Evaluation	Recall	Valence	Arousal
Success	Yes	Increase large	Increase large
Success	No	Increase small	Decrease small
Failure	Yes	Decrease large	Increase large
Failure	No	Decrease small	Decrease small

Arousal also follows a similar rule: it spikes after an instance-based prediction (large outcome change) and decays slowly during periods without such events. The specific increments and decrements applied to each parameter are summarized in Table 1.

Specifically, the following equation was used to update emotional valence (V) in equation 2.

$$V_{t+1} = V_t + \alpha(r - V_t) \tag{3}$$

r is set to 1 on success and -1 on failure. If the instance is used in each case, α is set to 0.2, otherwise α is set to 0.02. For arousal calculation, if instances are used, set r to 1 and α to 0.2. If no instances are used, the arousal level is calculated by setting r to -1 and α to 0.01.

Simulation

Aims and Settings

To observe how the emotions described in the previous section affect the process of instance-based intention estimation, we conduct a simulation involving two models. To examine the influence of emotions depending on the situation, we manipulate the model's heuristic (expertise) and the difficulty of the task as follows:

- Difficulty: Preliminary simulations using models that eliminate the influence of emotions have confirmed that the average game score can vary significantly depending on the starting deck arrangement. We ran 100 simulations using 10 randomly created initial deck arrangements, and the deck with the highest average score achieved a score of 16.0 points, while the deck with the lowest average score achieved a score of 11.3 points. The deck arrangement that achieved the highest average score was designated the "easy," and the deck arrangement that achieved the lowest average score was designated the "hard condition." The same initial arrangement was used for each trial for each difficulty level.
- Expertise: The decision heuristics are not given as the rules of Hanabi. Therefore, it seems to be acquired through repeated instance-based decision-making trials. Based on this idea, we set up novice and expert conditions by manipulating the heuristics. The novice condition does not take actions based on the partner's intentions, but only uses heuristics derived from the rules of the game. Specifically, it only uses "1. Play a confirmed card", "2. Hint for playable cards", and "4. Discard a confirmed card". In contrast, the expert condition makes decisions using all the heuristics 1 to 6.

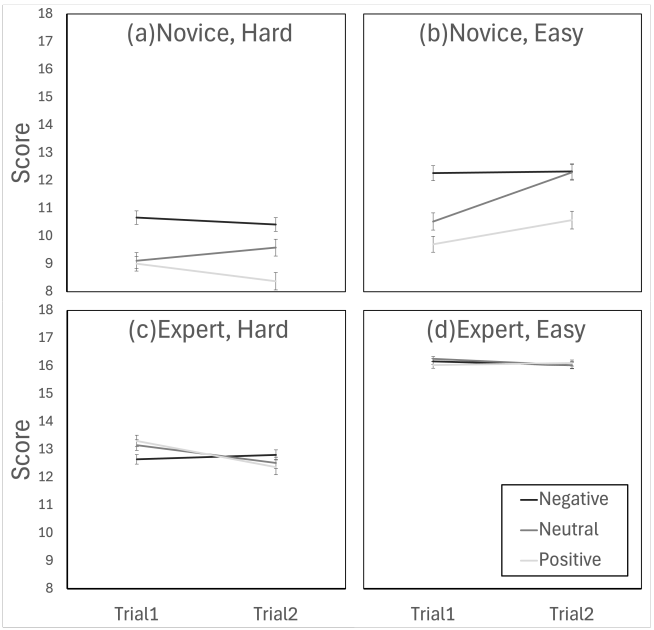


Figure 2: Simulation 1:Score of fixed emotion

Simulation 1: Fixed emotions

As a preliminary study to examine the role of emotions in intention estimation, we conducted simulations with fixed emotion parameters. The model that does not include equation 3 shown in the previous section recalls instances according to emotion parameters that are fixed to initial values.

Procedure In one run, the two models (with the same settings) performed two consecutive trials, with one trial being from the start to the end of the game. During the consecutive trials, the model memorized and used instances. We set "negative condition (arousal 0, valence -1)," "neutral condition (arousal 0, valence 0)," and "positive condition (arousal 0, valence 1)," and repeated 100 runs for each difficulty and expertise condition.

We assumed that different emotional states affect the recollection of experience used for intention estimation. In the negative condition, activations with failure instances are high, so failure instances are easily recalled, and it is difficult to make decisions about actions using experience. In the positive condition, the activations with success instances are high and they are easily recalled, so more decisions about actions are made using experience. In the neutral condition, there are no differences of activations between failure and success instances, so the frequency of use of experience is intermediate. Thus, the frequency of decisions about actions based on instances changes depending on the emotional condition. Based on such assumptions, this simulation aimed to explore emotional parameters suitable for cooperation in this task.

Results The results are shown in Figure 2. The vertical axis of each graph shows the average score of 100 runs and its standard error.

The influence of emotion on the score was significantly different between the novice and expert conditions. In the novice conditions, the highest score was obtained in the negative condition, and the lowest score in the positive condition. In addition, no variation in the scores between the trials was observed in the negative condition, but was observed in the neutral and positive conditions. In the novice-hard-positive condition, the score decreased due to learning (accumulation of instances), while in the novice-hard-neutral, novice-easy-neutral, and novice-easy-positive conditions, the score increased. On the other hand, in the expert condition, there was almost no difference between the trials or the influence of emotion. Among them, the positive and neutral conditions in the expert-hard condition showed a decrease in the scores due to learning, similar to the novice-hard-positive condition.

The reason why no learning effect was observed in the expert condition is thought to be because the behavior of the expert is largely fixed by heuristics. It is thought that the decisions of the expert are optimized for successful cooperation, and there is no room left for improving cooperation by utilizing experience. Therefore, in the hard conditions, where it is difficult to apply heuristics and where opportunities for instance-based decision-making increase, it is believed that a decrease in the scores due to learning was observed in all conditions other than the negative condition, which reduces the activation value of successful instances.

The decrease in performance shown in the above-mentioned expert-hard conditions was also observed in the novice-hard-positive condition. The negative effect of using instances in the hard conditions can be explained from the perspective of risk-accepting behavior. The positive condition reduces the penalty for partial matches in recalling successful instances, and actions are taken based on instances, even if there is a slight difference from the remembered instances. In the hard conditions, where the success rate of actions is low to begin with, risk-accepting behavior is thought to lead to a decrease in the scores. On the other hand, it is thought that the easy condition was an environment in which risk-accepting behavior was more likely to be successful than the hard conditions.

Simulation 2: Fluctuating emotion

We examined whether the negative effects of emotions observed in Simulation 1 could be improved by incorporating a mechanism for emotion fluctuation. By including equation 3 in the model, emotion parameters fluctuate depending on whether an action is successful or not. We confirmed whether recall using the fluctuating emotions would result in recall adapted to the situation.

Procedure As in Simulation 1, in one run, the two models with the same settings performed two consecutive trials, where the model accumulated and used instances. With arousal 0 and valence 0 as initial values, an "emotion change condition" was set in which the emotion parameters change according to equation 3, and 100 runs were repeated for each

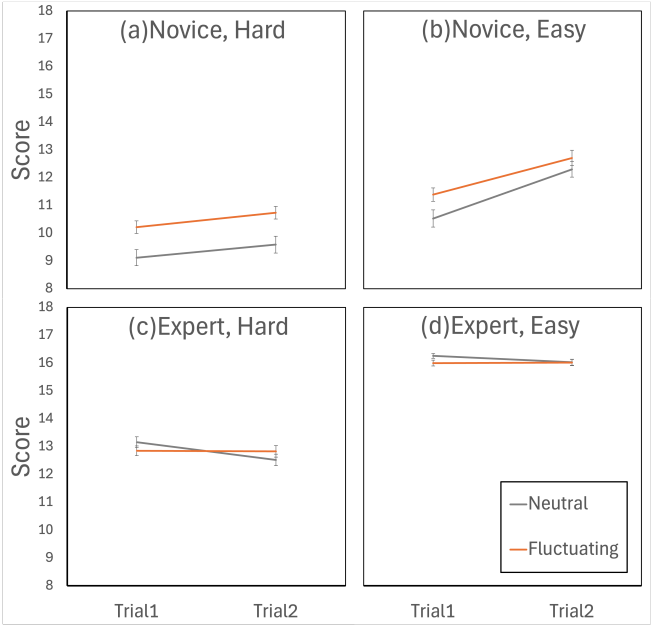


Figure 3: Simulation 2:Score of Fluctuating emotion

difficulty and expertise condition.

The recall of experiences used for intention estimation changes as emotions change. Since the activations of the success and failure instances changes according to the valence values, behaviors that utilize experiences decrease with a decrease in the valence, and increase with an increase in the valence. Contrary, the arousal value affects the probability of recalling instances regardless of the outcome. When the arousal is low, instances tends not to be recalled. Therefore, a decrease in the arousal leads to a decrease in instance-based decision, where as an increase in the arousal leads to an increase-based decision. As observed in Simulation 1, there is an appropriate valence according to the environment depending on the risk perspective, so it is thought that the score increases by adjusting the valence.

Results We compared the results of the emotion fluctuation condition and the neutral condition in Simulation 1. Figure 3 shows the change in the scores by the condition.

In the novice condition, the emotional fluctuation condition outperformed the neutral condition in both the hard and easy conditions. The effect of learning was observed as an increase in the scores, as in the neutral condition. On the other hand, in the expert model, no significant difference in the scores was observed between the two conditions. It is noteworthy that the decrease in the scores in the neutral condition, which was observed in the expert and hard conditions, was suppressed.

The fact that the score in the emotional fluctuation condition in the novice model exceeded that of the neutral condition indicates that the adjustment of emotional parameters according to the situation worked effectively. This effect of emotion regulation was also observed in the expert condition in the form of suppression of the negative effects of using

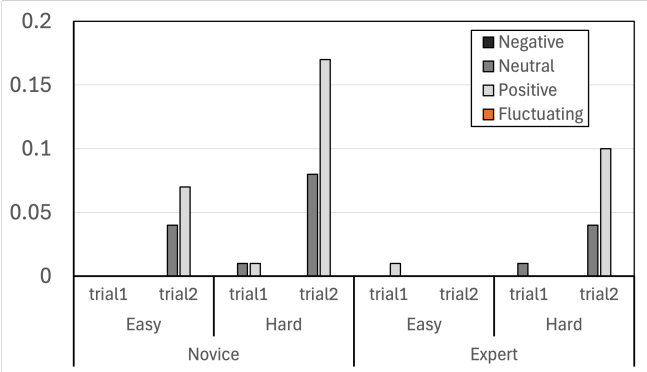


Figure 4: Ending a game due to failure to play card

experience.

Risk taking in the two simulations

To confirm whether the decrease of the score was caused by a risk taking, we classified the ending of the trials. Especially we focus on the ending caused by gaining three red tokens as risk taking behavior. Figure 4 shows the occurrence rate of ending the game due to gaining three rad tokens in the two simulations.

Regardless of expertise or difficulty, no ending due to three red tokes occurred in the negative and emotional fluctuation conditions. Furthermore, there were almost no gaining three red tokens in pre-learning (trial 1), and more occurred in post-learning (trial 2). When focusing on trial 2, the hard, novice, and positive conditions tended to gain more red tokens than the expert-negative-easy condition.

The difference between emotions in tendencies to gain red tokens is consistent with considerations from the perspective of risk management. The positive model is risk-accepting, which results in more gaining red tokens. This failure due to risk acceptance was even more noticeable in the hard conditions, where the success rate of the action was lower. Under the hard conditions, gaining red tokens was observed even in the expert condition. It is considered that under the hard conditions, there are many situations where any heuristics cannot be applied. In such situations, although the probability of success is low, risk-tolerant models played a card and resulted in gaining the red tokens.

Summary and Future Works

In this study, we assumed that emotions affect risk tendency in intention estimation, which affects the outcome of cooperative behavior, and constructed a Hanabi model that uses emotions for recall. A model corresponding to a novice with few available heuristics changed cooperative behavior by using emotions for recall, leading to successful cooperation when the emotional state matched the environment and failure when the emotional state was not suitable for the environment. On the other hand, a model corresponding to an expert with a lot of knowledge had relatively small effects of emotions because recall did not occur often, and the effects of emotions were no longer useful. It was also observed that the emotional state was adjusted to an appropriate state by incorporating emotional fluctuations into the model. The novice models with fluctuating emotions were more likely to succeed in collaboration by adopting appropriate emotions. In the expert models, the effect of fluctuating emotions was seen in the form of reduced failures.

The above results indicate that the function of emotions in decision-making changes depending on the situation, such as the difficulty of the task and the level of expertise. The adjustment of emotional states according to the difficulty of the task has long been studied from the perspectives of optimal arousal level theory (Yerkes & Dodson, 1908). In this study, we focused on emotional valence rather than arousal level, and suggested that there is an optimal emotional valence according to the difficulty of the task. In addition, the change in the influence of emotions due to task expertise is consistent with recent research on emotions in the field of cognitive modeling. Conway-Smith and West (2024) proposes a model in which the emotions associated with a similar experience are compiled into procedural memory as similar experiences are repeated, reducing the influence of emotions on cognitive tasks.

The model of this study confirmed that emotions reduce failure in cooperative behavior by adapting risk tendencies to the environment (hard/easy). Although the suggestion that emotions assist intention estimation was obtained, a direct relationship between emotions and intention estimation could not be confirmed. Although risk judgment was made by adjusting emotions, it was unclear whether the success of intention estimation was due to socialization. An example of socialization through emotions is the creation of an atmosphere due to a common emotional state. In the simulation of this study, the two models had the same emotional parameters and the same fluctuation structure. In order to examine the effect of a common emotional state on socialization and intention estimation, it is necessary to simulate whether cooperative behavior is successful when emotions are different.

Furthermore, in the future, it is necessary to develop the results obtained in this study and consider a more complete model of emotions and the success of intention estimation. To do this, it is necessary to consider not only the conditions between models, but also the learning process that leads to the acquisition of heuristics that were not included in the model of this study. In addition, in order to acquire production rules that are effective for the success of such intention estimation through experience accumulation, it will be necessary to incorporate learning from failure instances.

References

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

Bower, G. H. (1981). Mood and memory. *American Psychologist*, *36*(2), 129.

Conway-Smith, B., & West, R. L. (2024). The computational mechanisms of detached mindfulness. *ArXiv Preprint ArXiv:2409.15289*.

Cosmides, L., & Tooby, J. (1997). *Evolutionary psychology: A primer* (Vol. 13). Citeseer.

Czikszentmihalyi, M. (1990). *Flow: The psychology of optimal experience*. New York: Harper & Row.

Kawaji, R., Morita, J., & Osawa, H. (2024). Understanding emotion and emotional contagion effects on cooperative behavior through game simulation. In *Virtual mathpsych/iccm 2024*.

Kuwabara, R., Nagashima, K., Morita, J., Miyata, K., Kawagoe, A., & Osawa, H. (2023, may). Constructing a communication model for inference using the cooperative game hanabi. In *Human-agent interaction symposium 2023*.

Miyata, K., & Osawa, H. (2024). Enhancing cooperative behavior through subtle communication cues: The impact of hint order and hesitation in hanabi. In *Proceedings of the 12th international conference on human-agent interaction* (pp. 444–446).

Morita, J., Konnno, T., Okuda, J., Samejima, K., Li, G., Fujiwara, M., & Hashimoto, T. (2018). Cognitive factors influencing formation of collaborative communication -simmulation studies based on cognitive architecture-. *Paper of Human Interface Society*, *20*(4), 435–446.

Osawa, H. (2015). Estimation of own state by opponent's behavior in cooperative game hanabi. In *The 29th annual conference of the japanese society for artificial intelligence (2015)* (pp. 1F23–1F23).

Premack, D., & Woodruff, G. (1978). Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, *1*(4), 515–526.

Russell, J. A. (1980). A circumplex model of affect. *Journal of personality and social psychology*, *39*(6), 1161.

Turner, N. E., Zangeneh, M., & Littman-Sharp, N. (2006). The experience of gambling and its role in problem gambling. *International Gambling Studies*, *6*(2), 237–266.

Yerkes, R. M., & Dodson, J. D. (1908). The relation of strength of stimulus to rapidity of habit-formation. *Journal of comparative neurology and psychology*, *18*(5), 459–482.

Yuen, K. S., & Lee, T. M. (2003). Could mood state affect risk-taking decisions? *Journal of affective disorders*, *75*(1), 11–18.

Proceedings of the 23rd International Conference on Cognitive Modelling (ICCM 2025)

Estimating Emotion Related Parameters for Inter- and Intra-Individual Variability

Kohei Shimbori (shimbori.kohei.21@shizuoka.ac.jp)
Masaru Shirasuna (shirasuna-m@inf.shizuoka.ac.jp)
Junya Morita (j-morita@inf.shizuoka.ac.jp)
Department of Informatics, Graduate School of Integrated Science and Technology, Shizuoka University,
3-5-1, Johoku, Chuo-ku, Hamamatsu, Shizuoka, 432-8011, Japan

Abstract

Recent cognitive modeling studies have attempted to estimate individual parameters from human behavioral data. Such efforts are valuable for predicting performance on tasks and for understanding individual internal mechanisms. However, previous studies have primarily focused on estimating stable cognitive parameters such as processing speed and memory decay. In contrast, this study focuses on emotional parameters with large inter- and intra-individual variability. This study estimated parameters predicting performance in a simple memory task to investigate such variability. Our analysis of inter-individual variability revealed correlations between participants' state anxiety and both the "mismatch penalty in retrieval" and the "activation threshold required for retrieval." For intra-individual variability, results suggested a nonlinear relationship between fluctuations in emotional valence within the task and the mismatch penalty in retrieval. By refining these findings, we aim to concretize the parameters related to emotions within cognitive architectures.

Keywords: ACT-R; Emotion

Introduction

In cognitive science, computational models approximating human cognition have long been a central subject. A recent advancement involves estimating model parameters from individual human data (Kangasrääsiö, Jokinen, Oulasvirta, Howes, & Kaski, 2019). Models tailored to individuals can predict performance on arbitrary cognitive tasks. Furthermore, such a model leads to an understanding of individuals' specific behaviors as structured objective parameter sets. We believe that such an endeavor ultimately connects to long-term planning towards the improvement of various cognitive functions.

However, previous studies have mainly targeted parameters directly related to cognitive task performance, such as processing speed (i.e., thinking speed) and memory decay (Daily, Lovett, & Reder, 2001; Sense, Meijer, & van Rijn, 2018; Yang, Sibert, & Stocco, 2024). In contrast, parameters related to emotions, which significantly impact cognition, have not been explored sufficiently. Emotion, or affect[1], fluctuates both between individuals and within individuals. Clarifying the relationship between emotion and cognition within cognitive models is expected not only to deepen our understanding of human internal mechanisms but also to provide insight into emotional regulation, leading to the improvement of daily communication and the clarification of emotion-related behaviors.

This study reports an online experiment that uses a simple memory task accompanied by emotion-arousing stimuli. Through this experiment, we collect instances of memory errors in various affective states from individuals with diverse emotional traits. Using these data, we examine cognitive model parameters involved in both inter- and intra-individual variations in emotion. Regarding inter-individual variation, we investigate how cognitive model parameters estimated for each individual correspond to their attributes and subjective emotional states. For intra-individual variation, we analyze the correspondence between cognitive parameters and data segmented by emotion ratings obtained during the experiment.

Related Studies

In order to clarify the position of this research, we briefly introduce the cognitive modeling and emotional modeling approaches that this research relies on.

Cognitive Model and Cognitive Architecture

General cognitive architectures are useful in defining a parameter set that fits an individual. Among various architectures, ACT-R (Adaptive Control of Thought-Rational; Anderson, 2007) is widely recognized as a representative framework and has been extensively used in modeling diverse cognitive processes such as memory, attention, and decision-making.

Cognitive architectures provide standardized functions and parameter sets for simulating human cognitive processes. By adjusting these parameters according to specific tasks and individuals, predictions of concrete behaviors can be derived (Ueda et al., 2022; Stocco et al., 2024). However, a standardized methodology for parameter estimation has yet to be established.

In contrast, Shimbori et al. (2024) employed a grid search approach to estimate individual parameters by maximizing data fitting within a discretized parameter space. Notably, they pre-adjusted the range of parameters used in the grid search as its meta-parameters to maximize the overall fit to the experimental data. The exploration of these meta-parameters utilizes an algorithm belonging to the gradient descent. Their approach aims to estimate model parameters

[1] In this study, these terms are used interchangeably.

for each individual efficiently and with high precision. It is particularly effective in constructing models with large-scale data and high-dimensional parameter spaces.

Cognitive Model and Emotion

The interaction between cognition and emotion has been studied across various fields for a long time. Well-known theories in this area include the somatic marker hypothesis (Damasio, 1994) and the mood congruent effect (Bower, 1981). The former indicates a strong connection between emotion and decision-making, while the latter explains that positive memories are evoked in a positive mood and negative memories are evoked in a negative mood. As shown in these theories, cognition and emotion interact in complex ways and manifest in various behaviors. To understand this process, cognitive models have been employed in many research topics in this field.

Recently, discussions have emerged about incorporating an emotion module into the concept of a common model of cognition (Laird, Lebiere, & Rosenbloom, 2017), leading to advancements in research aiming to integrate emotion and cognition (Juvina, Larue, & Hough, 2018; Rosenbloom et al., 2024). In these studies, approaches have been taken to expand existing cognitive architectures, such as ACT-R, by adding new modules to represent emotion.

In contrast to the above approaches, the current study aims to examine the correspondence between existing parameters of ACT-R and emotion, with the goal of clarifying which emotions can be represented within the current ACT-R framework and identifying the parameters required for future expansions of the architecture. This approach aligns with the past idea of viewing emotion as a cognitive modulator to adapt to the situation (Ritter, 2009).

Experiment

This study uses a simple memory task to estimate cognitive parameters related to emotion. Memory is a cognitive function utilized across a wide range of cognitive tasks. Moreover, memory is closely related to emotion, and as discussed earlier, various investigations into their interaction have been conducted.

It is reasonable to assume that emotion / affect influences on memory errors, which are believed to be mainly caused by several biases, and include two main types as commission errors, where incorrect memories are recalled, and omission errors, where required memories cannot be recalled (Schacter, 1999). This study employs a model simulating these errors to fit human memory tasks. Specifically, a simple digits recall task was employed because of its ease of modeling and difficulty adjustment. In this task, participants are presented with a randomly generated number sequence within a specified time frame. Once the presentation of the sequence is completed, participants are required to report the sequence within a designated time.

In a preliminary experiment using this task, the variability of participants' emotional states was found to be small. To investigate the influence of emotion in more detail, the current study introduces manipulations designed to increase the emotional variability between individuals. This experiment was conducted with approval from the ethics committee of Shizuoka University.

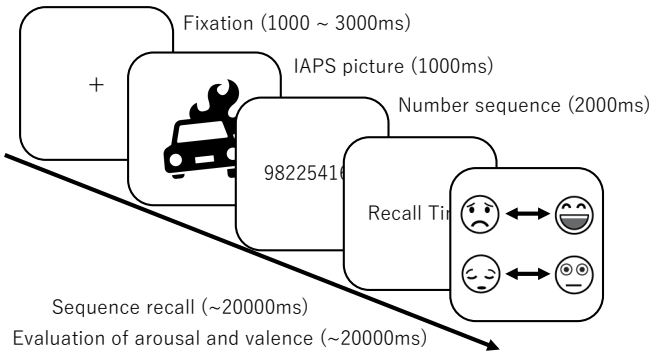


Figure 1: Digits recall task

Participants

We recruited 100 participants through a crowdsourcing site (Lancers.jp) and paid them 400 yen as compensation.

Materials

In the digits recall task, participants were presented with a number sequence on the monitor for 2 seconds, then entered the memorized sequence into a text box on the monitor using a keyboard within a 20-second response time. Once the response for one trial was completed, participants could immediately move to the next trial by pressing the enter key or the "End Input" button on the monitor. To manipulate the participants' emotional state, emotion-arousing stimuli were introduced before the sequence presentation. Additionally, to measure changes in emotional state during the task, subjective ratings of arousal and emotional valence were collected after each trial using the Affective Slider (Betella & Verschure, 2016). The flow of a single trial is illustrated in Figure 1.

The International Affective Picture System (IAPS; Lang, Bradley & Cuthbert, 2008) was used to induce emotional fluctuations. This dataset, designed and developed by the National Institute of Mental Health (NIMH) in the United States, is a standard dataset for studies on emotion and attention and contains approximately 1,000 color photographs. In this experiment, participants were randomly shown 20 positive and high-arousal images ($M_{valence} = 7.57$, $SD_{valence} = 1.63$, $M_{arousal} = 6.66$, $SD_{arousal} = 2.17$) and 20 negative and high-arousal images ($M_{valence} = 1.62$, $SD_{valence} = 1.18$, $M_{arousal} = 6.96$, $SD_{arousal} = 2.18$).

To obtain participants' personal attributes and emotional states during the task, we conducted pre- and post-task questionnaires. For the pre-task questionnaire, we measured the Japanese versions of the Positive and Negative Affect Schedule (PANAS; Watson, Clark & Tellegen, 1988; Sato & Ya-

suda, 2001; Kawahito, Otsuka, Kaida, & Nakata, 2011) and the state anxiety scale of the State-Trait Anxiety Inventory (STAI; Spielberger, Gorsuch, & Lushene, 1970; Nakazato & Mizuguchi, 1982). The post-task questionnaire included the same measures as the pre-task questionnaire, with the addition of the trait anxiety items from STAI. Furthermore, we collected personal attributes such as age, gender, and highest level of education, as well as subjective ratings of physical condition (1 = bad to 5 = good) and self-assessed memory ability (1 = very bad to 6 = very good). To exclude inattentive participants, dummy questions were embedded in the PANAS and STAI scales. Participants were instructed to select specific responses for the dummy items "Tired" and "Not Enjoying".

Procedure

Participants accessed the dedicated site via the link provided in the crowdsourcing site. They first reviewed a consent form for the emotion-arousing stimuli and confirmed their participation. After agreeing, they proceeded to an explanation of the experiment flow, entered a user ID, and completed the pre-task survey. Next, they worked on the 40 trials of the digits recall task, followed by the post-task survey. Finally, participants entered the displayed ID into the text box at the bottom of the crowdsourcing task page to finish the experiment.

Analysis 1: Inter-Individual Variation

Using the data obtained in the above experiment, we made two analyses, corresponding to the inter and intra-individual variability of emotion. In Analysis 1, parameters for each individual were estimated, and then the correlations between these and the participants' emotional ratings were examined. The following outlines the models used and the method of estimating parameters from the data.

Model

We used the grouped model from Tutorial Unit 5 (Bothell, 2022) in ACT-R version 7.27.9. This model divides a sequence of numbers into groups and stores each number based on its position within the group and the position of the group itself. Specifically, a sequence such as "123456789" is divided into groups like (123) (456) (789), where "1" is remembered as "the first digit in the first group." This memory structure derives from one in a past serial position memory model (Anderson, Bothell, Lebiere, & Matessa, 1998).

When recalling digits from memory, the model moves its attention focus from left to right within a group or between multiple groups. Initially, the "first digit of the first group" is retrieved, followed by the retrieval of other digits in order, attempting to recall the entire sequence.

When recalling numbers at each position in the sequence, the activation value of all digits in memory is computed, and the digit with the highest activation value is selected. In ACT-R, the activation value A_i of digit i is calculated as follows:

$$A_i = \sum_l PM_{li} + \varepsilon \quad (1)$$

where l distinguishes the retrieval requests, which include "group position" and "position within the group." M_{li} represents the similarity between the conditions and the target; a value of 0.5 for adjacent digits and groups with a weight P, which corresponds to a global parameter *mp* (mismatch penalty). Additionally, incorrect digit recall arises due to a temporal noise component, ε, corresponding to the global parameter *ans* (activation noise s). Thus, the smaller the *mp* and the larger the *ans*, commission error is more likely to happen.

On the other hand, omission errors occur when the activation values of all memory items fall below a threshold determined by the *rt* (retrieval threshold) parameter. In other words, the larger the value of *rt*, the less likely it is that a digit will be recalled.

Parameter Fitting

Index This study assesses the fitting between the experimental data and the model output based on Histogram Intersection (HI; Swain & Ballard, 1991) of edit distance[2] between the recalled and the presented number sequences. Since the frequencies were different between the experimental data and the model output, they were normalized so that the sum of the frequencies was 1 for the comparison.

Fitting process To reproduce the commission and omission errors in the responses, the *mp* and *rt* that best fit the individual experimental data are searched. Since there are countless appropriate ranges and combinations of parameters, we first roughly specify the range of parameters (meta parameters of grid search), and then search within that range as follows:

1. Determine the initial median and range of *mp* and *rt*.
2. Generates candidates with shifted medians and ranges for each parameter. The shifted median is the current value plus $+0.5$, $+0$, or -0.5, and the shifted range is the current value multiplied by $*1.2$, $*1.0$, or $*0.8$.
3. For each median and range combination (3^4), discretize the parameter values into 10 steps.
4. For each combination of *mp* and *rt*, the model is executed 100 times using 100 different parameter sets (10 steps of *mp* $\times$ 10 steps of *rt*) and construct a histogram of edit distances. In addition, a similar histogram was constructed for each participant's responses. For each participant's histogram, we identify the histogram of the model that best matches and compute the sum for all participants for that HI.
5. Execute the above 4 for 3^4 combinations of the currently set median values and ranges of *rt* and *mp*. Compare the sum of the largest HIs in that combination to the largest HI in the past median and range settings. If a value exceeding the sum of past HIs is obtained from the combination of

[2] python-Levenshtein package was used (`https://pypi.org/project/python-Levenshtein/`).

Table 1: Attributes and base performance of the task.

	Mean	Median	SD	Min	Max
Age	45.42	45.50	10.26	19.00	70.00
Physical Condition	3.65	4.00	1.01	1.00	5.00
Academic Background	3.60	4.00	0.85	2.00	6.00
Memory Ability	2.57	2.00	0.90	1.00	4.00
Positive (Pre)	30.50	32.00	9.27	10.00	50.00
Positive (Post)	25.38	24.50	9.99	10.00	50.00
Positive (Post - Pre)	-5.13	-5.00	7.65	-25.00	15.00
Negative (Pre)	17.64	15.50	7.21	10.00	46.00
Negative (Post)	33.75	35.00	10.71	10.00	51.00
Negative (Post - Pre)	16.11	16.00	9.56	-6.00	40.00
State Anxiety (Pre)	44.94	45.00	9.47	25.00	67.00
State Anxiety (Post)	58.32	58.00	8.88	36.00	76.00
State Anxiety (Post - Pre)	13.38	12.00	9.98	-7.00	37.00
Trait Anxiety	48.93	46.50	12.57	23.00	75.00
Arousal	0.64	0.63	0.11	0.41	0.99
Valence	0.40	0.41	0.09	0.18	0.60
Edit Distance	2.19	2.23	0.90	0.05	4.35
Missing Digits	0.54	0.13	0.88	-0.23	3.55

the current median and range, those at that point are used as the value of the range for the next step.

6. Repeat the above until the maximum value of the sum of HI is no longer updated.

The above procedure identifies the parameter range that maximizes the fitting for the entire experimental data. Among them, the parameter with the highest HI for each individual is selected as the individual's parameter. However, if the agreement of the individual parameters obtained in the first stage (meta parameter search) is higher than those obtained in the second stage (grid search), the individual's parameters derived from the first stage are adopted.

Result

Data from 28 participants who failed the dummy questions were excluded, leaving a final sample of 72 participants (52 males, 20 females, $M_{age} = 45.42$, $SD_{age} = 10.26$). Table 1 presents an overview of the questionnaire results and task performance obtained from the participants.

Following the aforementioned procedure, individual ACT-R parameters were estimated by setting the initial values of the median and range for each parameter to 1.0. The model parameters corresponding to the highest HI across five estimation attempts were determined as individual parameters, and correlations between the obtained parameters and individual attributes or emotional scores were analyzed. Table 2 shows the fitting results, while Table 3 presents the correlation analysis results.

A significant correlation was observed between the pre-task state anxiety score and the model parameters *mp* and *rt* ($n = 72$, $p < .05$). Additionally, a significant positive correlation was found between the difference in positive affect scores before and after the task and the model parameter *mp* ($n = 72$, $p < .05$).

Table 2: Fitting result of experiment

	HI (%)	*mp* median	*mp* range	*rt* median	*rt* range	Updates
Trial 1	53.41	1.50	0.96	-1.00	1.15	4
Trial 2	67.47	1.00	0.13	-0.50	0.12	14
Trial 3	67.06	1.00	0.17	-0.50	0.19	11
Trial 4	67.39	1.00	0.13	-0.50	0.19	13
Trial 5	67.04	1.00	0.19	-0.5	0.16	12
Mean	64.47	1.10	0.32	-0.60	0.36	10.8
Median	67.05	1.00	0.18	-0.5	0.19	11.5
Max	67.47	1.50	0.96	-0.50	1.15	14
Min	53.41	1.00	0.13	-1.00	0.12	4

Table 3: Correlation analysis

	HI	*mp*	*rt*
Age	0.076	0.085	-0.159
Gender	0.069	0.047	-0.118
Academic Background	-0.035	-0.059	0.091
Physical Condition	-0.039	0.072	-0.111
Memory Ability	0.080	0.007	-0.030
Positive (Pre)	-0.006	-0.218	0.213
Positive (Post)	-0.049	-0.020	0.075
Positive (Post - Pre)	-0.057	0.237*	-0.161
Negative (Pre)	-0.158	0.133	-0.136
Negative (Post)	0.075	0.082	-0.126
Negative (Post - Pre)	0.203	-0.009	-0.039
State Anxiety (Pre)	0.065	0.305**	-0.287*
State Anxiety (Post)	0.134	0.076	-0.103
State Anxiety (Post - Pre)	0.058	-0.222	0.181
Trait Anxiety	0.196	0.230	-0.061
Arousal	-0.013	-0.079	0.097
Valence	-0.007	0.053	-0.065
Edit Distance	-0.616*	-0.511***	0.233*
Missing Digits	-0.331*	-0.044	0.078

$*p < .05, **p < .01, ***p < .001$

Discussion

The results indicated a tendency for higher state anxiety to be associated with higher *mp* and lower *rt*. This suggests that individuals with higher state anxiety exhibit a tendency to suppress both commission and omission errors. Alternatively, as shown in Table 1, negative affect scores and state anxiety increased before and after the task. This implies that the presentation of images may have induced anxiety, leading participants to respond more cautiously.

Participants who exhibited a greater decrease in positive affect scores also tended to have lower *mp*. Focusing on the change in positive affect scores in Table 1, the average decrease among participants was 5.13 points, suggesting that the image presentation under the current experimental conditions led to a decline in positive emotions. This result implies that participants who exhibited a stronger tendency to commit commission errors also experienced a greater reduction in positive emotions. Conversely, it could also be interpreted that the decrease in positive emotions enhanced the tendency to produce commission errors.

Additionally, significant correlations were observed between the mean edit distance and each parameter, indicating

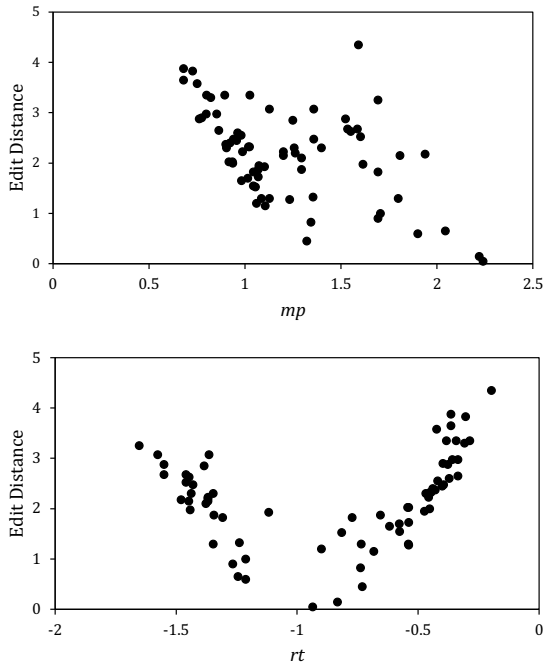


Figure 2: Relationship between edit distance and parameters

Table 4: Fitting results grouped by arousal

Arousal	HI (%)	*mp*	*rt*	valence	edit distance	missing digits	n
0.00 - 0.09	82.15	1.22	-0.47	0.86	2.18	0.74	66
0.10 - 0.19	84.75	1.42	-1.17	0.67	2.16	0.65	80
0.20 - 0.29	87.10	1.63	-1.36	0.67	2.17	0.36	126
0.30 - 0.39	84.11	1.22	-0.47	0.58	2.10	0.46	202
0.40 - 0.49	83.46	1.63	-1.36	0.50	2.05	0.39	183
0.50 - 0.59	83.12	1.70	-1.32	0.39	2.29	0.67	410
0.60 - 0.69	84.72	1.22	-0.47	0.40	2.16	0.50	526
0.70 - 0.79	82.62	1.22	-0.47	0.37	2.23	0.63	494
0.80 - 0.89	82.05	1.22	-0.47	0.27	2.21	0.68	349
0.90 - 1.00	82.78	1.41	-0.97	0.20	2.14	0.39	431

Table 5: Fitting results grouped by emotional valence

Valence	HI (%)	*mp*	*rt*	arousal	edit distance	missing digits	n
0.00 - 0.09	86.69	1.55	-1.34	0.82	2.32	0.63	714
0.10 - 0.19	83.03	1.84	-1.45	0.70	2.29	0.46	289
0.20 - 0.29	84.22	1.78	-0.40	0.68	2.19	0.76	259
0.30 - 0.39	83.44	1.78	-0.40	0.62	2.00	0.47	259
0.40 - 0.49	85.77	2.19	-1.12	0.56	2.38	0.57	183
0.50 - 0.59	87.99	1.85	-1.44	0.55	2.14	0.50	239
0.60 - 0.69	82.41	1.84	-1.45	0.53	2.03	0.49	317
0.70 - 0.79	82.42	1.55	-1.34	0.53	2.16	0.58	240
0.80 - 0.89	82.80	1.42	-1.22	0.52	2.05	0.47	186
0.90 - 1.00	86.36	1.55	-1.34	0.54	2.03	0.30	181

higher *mp* and lower *rt* minimize errors. However, when we look at scatter plots for these correlations (Figure 2), more complex figures are observed; the edit distance was lowest around $rt = -1$, increasing when *rt* was either extremely high or low. This result suggests the presence of a nonlinear relationship that cannot be explained by a simple correlation, and further detailed analysis is required.

Analysis 2: Intra-Individual Variation

Contrary to analysis 1, where the relationship between individual emotional traits and cognitive parameters was examined, Analysis 2 explores cognitive parameters associated with emotional fluctuations during the task.

Parameter Fitting

In the task, subjective ratings of arousal and emotional valence were recorded on a scale from 0 to 1, and the trials were grouped into ten bins based on these ratings. For each bin, the *mp* and *rt* parameters were estimated using the same fitting procedure as in Analysis 1, treating each bin as an individual participant. After five iterations of the fitting process, the parameters that yielded the highest HI were selected for each bin.

Result

Tables 4 and 5 present the goodness-of-fit (HI) and estimated parameter values (*mp*, *rt*) for each bin of arousal and valence. Additionally, the tables include the number of data points in each bin and task performance metrics. These values indicate that the estimation for both arousal and valence achieved a higher HI compared to Table 2[3]. However, it is not straightforward to find relationships between bin values and specific performance metrics.

Figure 3 illustrates the relationship between the rated affects and the estimated parameters. For arousal, no clear systematic relationship with *mp* or *rt* was observed. However, the valence graph appears to exhibit an inverted U-shaped pattern for both *mp* and *rt*, albeit with different peak positions. This suggests a nonlinear relationship between emotion and cognitive parameters, wherein *mp* and *rt* are highest when valence is neutral, and both parameters decrease when valence is either extremely high or low.

Discussion

In Analysis 2, parameters of the cognitive model were estimated based on data segmentation using affective ratings without considering individual differences. The results indicate that while the data-to-model fitting was not low, no easily interpretable relationship was found between affective ratings and the estimated parameters. This result suggests that the influence of long-term emotional variations (individual traits) on cognitive parameters may be greater than that of temporary emotional fluctuations.

Although issues remain regarding the reliability of the results, the inverted U-shaped relationship between affective values and cognitive parameters is observed in Figure 3. This result aligns with findings frequently reported in emotion-

[3]However, the HI values in Table 2 are computed from the sum of the meta-parameter search results rather than directly obtained from the grid search.

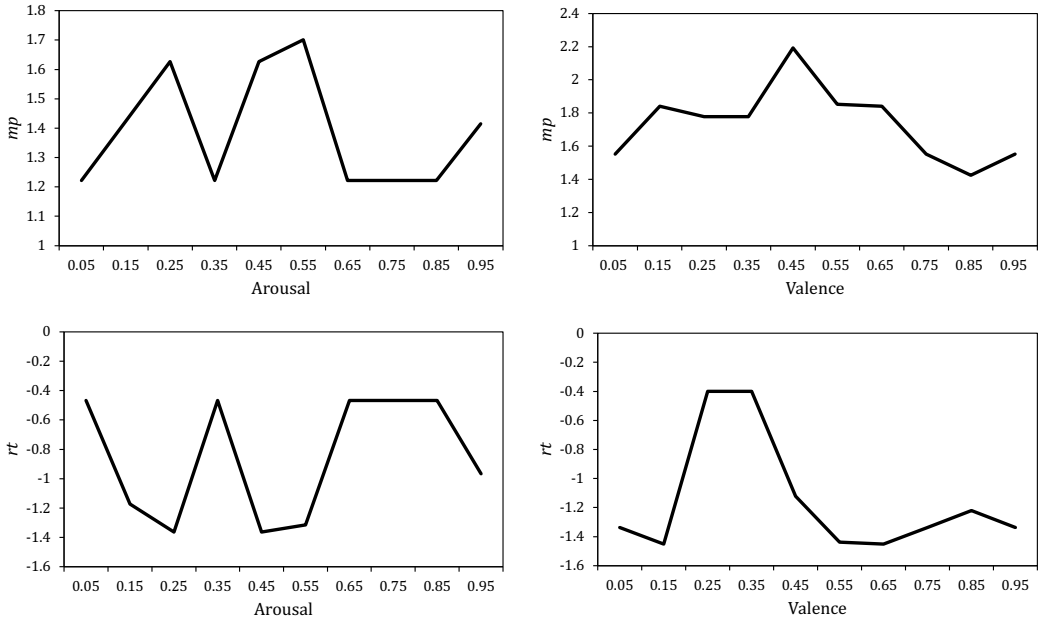


Figure 3: Distribution of *mp* and *rt* by arousal and valence

related research (Yerkes & Dodson, 1908; Csikszentmihalyi, 1990), suggesting that when emotional valence is in extreme states, memory-related parameters shift toward values that allow for more errors, increasing the likelihood of both commission and omission errors. Further robust analyses will be necessary to verify this interpretation in future research.

Conclusion

This study explored the possibility of estimating emotions as parameters in ACT-R using behavioral data, focusing on both inter-individual and intra-individual variations. We estimated individual cognitive parameters through a two-step process: first, dynamically narrowing the search range based on overall response data, and then performing a grid search within this range to identify the best-fitting parameters. During the narrowing, parameters yielding improved model fit were selected as the individual's estimates. Using these parameters, we examined their relationships with affective rating scales and anxiety measures. As a result, correlations were found between subjective ratings related to emotions and anxiety and parameters associated with omission and commission memory errors. Furthermore, when the data were segmented by arousal levels and emotional valence, differences in parameter values were observed between neutral and non-neutral affective states. These findings suggest that emotions can be represented using existing ACT-R parameters while also highlighting the necessity of modeling intra-individual emotional fluctuations.

However, to lead a robust result of estimation, we need to improve fitting between the model and participants. The low absolute values of the correlations in Table 3 suggest that existing parameters alone may not sufficiently capture the diverse aspects of individuals. Given these results, future study may require new model parameters, or expanding the model itself as suggested by Rosenbloom et al. (2024).

By addressing these technical challenges step by step and clarifying cognitive parameters specific to individuals and emotions, our approach is expected to contribute to various practical applications in individual behavior simulation. The estimated parameters, such as *mp* and *rt*, are generalizable and can be applied to other tasks. Consequently, utilizing individual specific parameters may enable predictions of participants' behavior in tasks beyond the current study. In particular, support systems that incorporate models capable of externalizing internal states could be beneficial for individuals experiencing psychological stress or anxiety (LeDoux, 1998).

References

Anderson, J. R. (2007). *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.

Betella, A., & Verschure, P. F. M. J. (2016, 02). The affective slider: A digital self-assessment scale for the measurement of human emotions. *PLOS ONE*, *11*(2), 1-11.

Bothell, D. (2022). Unit 5: Activation and context, ACT-R tutorial [Computer software manual].

Bower, G. H. (1981). Mood and memory. *American Psychologist*, *36*(2), 129.

Csikszentmihalyi, M. (1990). *Flow: The psychology of optimal experience*. New York: Harper & Row.

Daily, L., Lovett, M., & Reder, L. (2001). Modeling individual differences in working memory performance: A source activation account. *Cognitive Science*, *25*, 315-353.

Damasio, A. R. (1994). *Descartes' error: Emotion, reason, and the human brain*. Putnam.

Juvina, I., Larue, O., & Hough, A. (2018). Modeling valuation and core affect in a cognitive architecture: The impact of valence and arousal on memory and decision-making. *Cognitive Systems Research*, *48*, 4–24.

Kangasrääsiö, A., Jokinen, J. P. P., Oulasvirta, A., Howes, A., & Kaski, S. (2019). Parameter inference for computational cognitive models with approximate bayesian computation. *Cognitive Science*, *43*(6), e12738.

Kawahito, J., Otsuka, Y., Kaida, K., & Nakata, A. (2011). Reliability and validity of the japanese version of 20-item positive and negative affect schedule. *Hiroshima Psychological Research*, *11*, 225–240.

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.

Lang, P., Bradley, M., & Cuthbert, B. (2008). *International affective picture system (IAPS): Affective ratings of pictures and instruction manual* (Technical Report No. A-8). Gainesville, FL: University of Florida.

LeDoux, J. E. (1998). *The emotional brain: The mysterious underpinnings of emotional life*. Simon and Schuster.

Nakazato, K., & Mizuguchi, T. (1982). Development and validation of japanese version of state-trait anxiety inventory : A study with female subjects. *Japanese Journal of Psychosomatic Medicine*, *22*(2), 107-112.

Ritter, F. E. (2009). Two cognitive modeling frontiers. *Information and Media Technologies*, *4*(1), 76-84.

Rosenbloom, P., Laird, J., Lebiere, C., Stocco, A., Granger, R., & Huyck, C. (2024). A proposal for extending the common model of cognition to emotion. In *Proceedings of the 22th international conference on cognitive modeling.*

Sato, A., & Yasuda, A. (2001). Development of the japanese version of positive and negative affect schedule (panas) scales. *The Japanese Journal of Personality*, *9*(2), 138-139.

Schacter, D. L. (1999). The seven sins of memory: Insights from psychology and cognitive neuroscience. *American Psychologist*, *54*(3), 182–203.

Sense, F., Meijer, R. R., & van Rijn, H. (2018). Exploration of the rate of forgetting as a domain-specific individual differences measure. *Frontiers in Education*, *3*.

Shimbori, K., Nishikawa, J., Nagashima, K., & Morita, J. (2024). *Investigating influential factors and internal interactions through parameter estimation on memory errors.* (Paper presented at Virtual MathPsych/ICCM 2024. Via mathpsych.org/presentation/1641)

Spielberger, C. D., Gorsuch, R. L., & Lushene, R. E. (1970). *Stai manual for the state-trait anxiety inventory ("self-evaluation questionnaire")*. Consulting Psychologists Press.

Stocco, A., Mitsopoulos, K., Yang, Y. C., Hake, H. S., Haile, T., Leonard, B., & Gluck, K. (2024). Fitting, evaluating, and comparing cognitive architecture models using likelihood: A primer with examples in ACT-R. *arXiv preprint arXiv:2410.18055v1*.

Swain, M. J., & Ballard, D. H. (1991). Color indexing. *International journal of computer vision*, *7*(1), 11–32.

Ueda, K., Sakamoto, R., Ishii, H., Shimoda, H., Obayashi, F., & Morita, J. (2022). Examining the mechanism of concentration on intellectual works by simulation using cognitive architecture. In *Intelligent human systems integration (ihsi 2022): Integrating people and intelligent systems.*

Watson, D., Clark, L. A., & Tellegen, A. (1988). Development and validation of brief measures of positive and negative affect: The panas scales. *Journal of Personality and Social Psychology*, *54*(6), 1063–1070.

Yang, Y., Sibert, C., & Stocco, A. (2024). Reliance on episodic vs. procedural systems in decision-making depends on individual differences in their relative neural efficiency. *Computational Brain & Behavior*, *7*(3), 420–436.

Yerkes, R. M., & Dodson, J. D. (1908). The relation of strength of stimulus to rapidity of habit-formation. *Journal of Comparative Neurology and Psychology*, *18*(5), 459–482.

Visions remembered—Using a Vision-Language Model to set and recall Image Impressions in the Memory of a Cognitive Model

Thomas Sievers (t.sievers@uni-luebeck.de)
Nele Russwinkel (nele.russwinkel@uni-luebeck.de)
Institute of Information Systems University of Lübeck
Lübeck, Germany

***Abstract*—Large Language Models (LLMs) and Vision-Language Models (VLMs) have the potential to emulate human cognitive abilities for robots and thus significantly advance the evolution and use of cognitive architectures. This opens up opportunities for using human-like judgment and decision-making capabilities such as instance-based learning and intuitive decision making inherent in cognitive architectures with social robots. Using a combined system of an Adaptive Control of Thought-Rational (ACT-R) model and a humanoid social robot, we show how content from the declarative memory of the ACT-R model can be retrieved per association of real-world data obtained by the robot via the image recognition capabilities of an LLM. Such recollections can be fetched and processed according to the procedural memory of cognitive model productions and then returned to the robot as instructions, for example to add the prompt to LLM-driven utterances to keep them more contextual. In addition, visual impressions captured by the robot can be stored in the cognitive model.**

I. Introduction

Large Language Models (LLMs) like ChatGPT have advanced reasoning capabilities, blending intuitive and deliberate cognitive processes [6]. LLMs help robotic systems improve their generalization capabilities in dynamic and complex real-world environments and can significantly increase their behavior planning and execution capabilities, enabling robots to engage with their environment in a human-like manner [12].

Vision-Language Models (VLMs) are multimodal AI systems created by combining an LLM with a vision encoder that gives the LLM the ability to "see". They provide assistance with complex tasks such as creating captions and answering visual questions [7]. VLMs are capable of performing a variety of tasks after learning relationships between images and language from large data sets, such as answering questions about images, finding sentences that correspond well with images, and finding image regions that correspond to texts. These skills can be used in many ways for robotics, for example for robot movements, state recognition, object recognition, affordance recognition, relation recognition, and anomaly detection [14, 25].

Cognitive architectures refer both to a theory about the structure of the human mind and to a computer-based implementation of such a theory. Their formalized models can be used to react flexibly to actions in a human-like manner and – when used in a robot – to develop a situational understanding for adequate reactions. ACT-R (Adaptive Control of Thought - Rational) is a well-known and successfully used cognitive architecture [2]. A cognitive architecture can also be used to add a "human component" to robotic applications [29]. Furthermore, language models are good at fast automatic reasoning, but less capable of high-level cognition to enable complex mental operations and "slow thinking" following the dual process theory of human cognition [13].

Looking at Human-Robot Interaction (HRI), a combination of robot sensor technology and data processing with a cognitive architecture offers the possibility of dealing with information from the robot's real world in cognitive models. With their ability to use general concepts inspired by the human brain and create mental models based on human cognitive abilities, such architectures can help provide facts and context, e.g. in the form of memories from a particular scenario, for an LLM and provide individualized experiences. Using prompt augmentation, conclusions of a mental model can be taken into account in utterances of the LLM.

Intuitive decision-making as a subjective, particularly human type of decision-making, is based on implicit knowledge that is transmitted to the conscious mind at the time of the decision through affect or unconscious cognition. Computational models of intuitive decision-making can be expressed as instance-based learning using the ACT-R cognitive architecture [26]. In instance-based learning theory (IBLT), past experiences (i.e. instances) are retrieved using cognitive mechanisms of a cognitive architecture. IBLT proposes learning mechanisms related to a decision-making process, such as instance-based knowledge and recognition-based retrieval. These learning mechanisms can be implemented in an ACT-R model [9, 8]. In our opinion, it would be interesting to enable such behavior for a social robot as well.

We apply the LLM of OpenAI's Generative Pretrained Transformer (GPT, commonly known as ChatGPT) in an application example for a dialog between a human and a robot

using it for the robot's utterances [20]. Our application is primarily intended to demonstrate the technical possibility of realization. However, the proposed method can also be used for other scenarios and beyond an HRI context.

Visual impressions of the robot are processed by the LLM in such a way that the core content of what the robot sees is expressed in three keywords or key phrases. These keywords or phrases are passed to an ACT-R model as chunks and processed there. The cognitive model uses these chunks to search its declarative memory for existing memory content that is indexed in the same or a similar form using chunks. The stored memory content contains a recollection phrase that reflects the visual impression at the particular time in the form of a short, complete sentence. If there is a positive correlation between chunks from the LLM and memory chunks, this recollection or *fact* phrase is passed to the LLM for prompt augmentation, which generates a response based on this knowledge. This process can be seen as an association of remembered impressions. An a-priori factual knowledge can be accumulated by creating corresponding chunks in the declarative memory. This leads us to the following hypothesis:

Hypothesis Utilizing memory chunks from a cognitive model enables prompt augmentation for an LLM with associated past visual impressions.

Using ACT-R's chunk and memory system to store and process impressions of a certain scenario and enabling an LLM to incorporate remembered data from such impressions into utterances, opens a path for a more reliable and evidence-based application of LLMs. We provide an insight into our ongoing work and summarize initial findings.

II. Related work

Robots are able to use Large Multi-modal Models (LMMs) to comprehend and execute tasks based on natural language input and environmental cues, and the integration of foundation models such as LLMs and VLMs can effectively improve robot intelligence [18, 12, 28].

VLMs help to equip robots with the ability to physically-based task planning and can analyze videos of humans performing tasks to obtain a symbolic task plan for a robot through textual explanations of the environment and action details in combination with an LLM-based task planner [11, 27].

Yoshida et al. investigate possibilities for the development of a "minimal self" with a sense of agency and ownership in a robot that is able to mimic human movements and emotions by using human knowledge from language models [31]. To do this, they use GPT-4's motion generation and image recognition capabilities. VLMs as a basis for metacognitive thinking can enable robots to understand and improve their own processes, avoid hardware failures and thus increase their resilience [16].

Since the recent successes of language models, there has been an increased interest in the interplay between LLMs and cognitive architectures. Niu et al. provided an overview of similarities, differences, and challenges between LLMs and cognitive science by analyzing methods for assessing potential cognitive capabilities of LLMs, discussing biases and limitations, and an integration of LLM with cognitive architectures [19]. In the novel neuro-symbolic architecture presented by Wu et al., human-centered decision making was enabled through the integration of ACT-R with LLMs by using knowledge from the decision process of the cognitive model as neural representations in trainable layers of the LLM [30]. They found that this improved the ability for grounded decision making.

Knowles et al. proposed a system architecture that combined LLMs and cognitive architectures with an analogy to "fast" and "slow" thinking in human cognition [15]. Leivada et al. explored whether the current generation of LLMs is able to develop grounded cognition that incorporates prior expectations and prior world experiences to perceive the big picture [17]. Insights from human cognition and psychology anchored in cognitive architectures could contribute to the development of systems that are more powerful, reliable and human-like [24]. The dual-process architecture and the hybrid neuro-symbolic approach to overcoming the limitations of current LLMs is seen as particularly important for this.

A cognitive system based on ACT-R was used by Birlo et al. for their concept of robot self-awareness in an embodied robotic system [3]. They focused on the representation of internal states of the robot, which processed external states from the real world and created its own interpretation of what it perceived. Sievers et al. used a cognitive model to store and process word associations for the grounding of words in a language game between a robot and a human [23].

An example integration of an LLM with a cognitive architecture to enable planning and reasoning in autonomous robots was given by Gonzáles-Santamarta et. al [10]. They proposed how to leverage the reasoning capabilities of LLMs inside the ROS 2-integrated cognitive architecture MERLIN2. Sievers et al. give examples of a connection between a cognitive model and a robot application via a TCP/IP connection of the *dispatcher* included in the ACT-R software [21, 22].

III. Methods

For a human-like way of retrieving recollections from memory, we propose to use the declarative memory of a cognitive architecture in conjunction with procedural capabilities of a cognitive model. Our system for testing such an approach consisted of the humanoid social robot Pepper with a software application that connects to OpenAI's GPT-4o model via an Application Programming Interface (API), and a TCP/IP connection over a wireless network to the ACT-R 7 cognitive architecture software version 7.28 running on a remote computer.

ACT-R offers the technical possibility of integrating a cognitive model with bidirectional communication into an application of choice. To establish a remote connection from the robot application to ACT-R, we are using it's remote interface – the *dispatcher*. Fig. 1 shows the setup of the bidirectional connection between the dispatcher, which acts as a kind of server, and the client application of the robot.

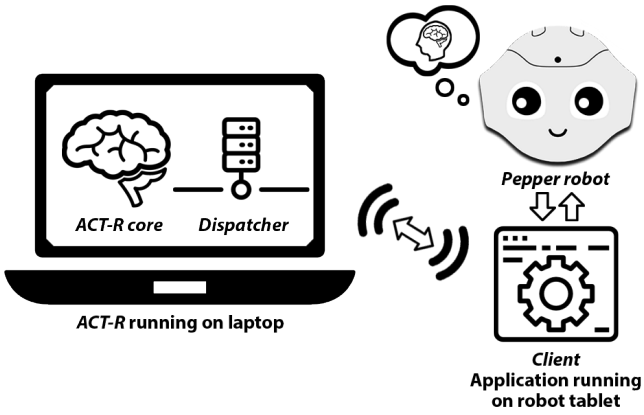


Fig. 1. Connection between ACT-R / Dispatcher and the robot application.

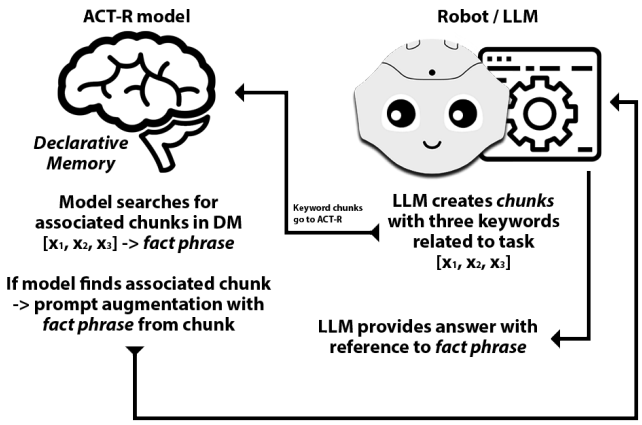


Fig. 2. Transfer of keywords to ACT-R and return of retrieved memory facts to the LLM, e.g. for prompt augmentation.

A. Cognitive Model

We used the standalone application of ACT-R 7, for which we created a cognitive model using LISP [5]. In ACT-R, declarative knowledge is represented in the form of chunks, i.e. representations of individual properties, each of which can be accessed via a labeled slot. The cognitive model programmed in LISP for our example usage should receive chunks with keywords from the robot application and checking whether chunks for these keywords are already stored in the declarative memory of the model in a suitable combination. We assumed the transfer of three keywords. The programmed productions of the procedural memory checked all combinations of the sequence of keywords for a match with memory content and generated a hit for two out of three matching keywords. In this case, the associated memory content including a corresponding recollection was called up and the recollection was returned to the robot application.

Our LISP code defined a chunk type for the memory content in declarative memory as follows: *(chunk-type keyword asso-one asso-two asso-three phrase)*

This chunk type 'keyword' featured three labeled slots 'asso-one', 'asso-two' and 'asso-three' as well as a 'phrase' slot, which stored the recollection we wanted to retrieve. For example, this could be a remembered fact used to complete a system prompt for an LLM that enables the robot to talk to humans. Each chunk stored in the declarative memory also has an arbitrary name. Figure 2 shows the process of searching for a matching memory chunk. Having found a corresponding memory chunk, the cognitive model returned the sentence stored in the 'phrase' slot of this chunk as a recollection or *fact phrase*. Example content for such a memory chunk could be: asso-one *person* asso-two *indoor* asso-three *computer* phrase *A person sitting in front of a computer indoors*

B. Humanoid Social Robot Pepper

The humanoid social robot Pepper, shown in Figure 3, was developed by Aldebaran and first released in 2015 [1]. The robot is 120 centimeters tall and optimized for human interaction. It is able to engage with people through conversation, gestures and its touch screen. Pepper is equipped with internal sensors, four directional microphones in his head and speakers for voice output. Speech recognition and dialog is available in 15 languages. The robot features an open and fully programmable platform so that developers can program their own applications using software development kits (SDKs) for programming languages like Python, Java or Kotlin.

In dialogs with humans, our robot application forwarded utterances of the human dialog partner as input to the OpenAI API, which returned a dictionary with the status and response of the API. With each API call, the entire dialog was transferred to the GPT model. This allows the model to constantly 'remember' what was previously said and refer to it as the dialog progressed. In this way, complex dialogs between humans and the robot became possible.

C. Retrieving recollections based on vision

The system's ability to retrieve recollections from declarative memory by recognizing visual impressions from the robot's perspective should be demonstrated. The LLM was instructed to create a description of the content in the form of three keywords about the content of an image taken with the robot's camera equipment. The robot's front camera took a picture every few seconds to capture an up-to-date impression of what the robot saw. To prepare a response to a human utterance based on visual impressions, the robot application sent the currently captured image to the GPT-4o model as a completion task, using the type *'image_url'* instead of *'text'*.

The three image-related keywords from the model's response were transmitted as chunks to the cognitive model to search for comparable chunks in declarative memory. In the event of a match, the content of the 'phrase' chunk slot was transferred to the robot application for further use.

In the context of our example application we focused on possible uses of recollections from declarative memory for prompt augmentation of a GPT model. But the other way around, it is also possible with this system to write memory chunks and thus visual impressions, which are provided with

Fig. 3. Robot Pepper shows a picture of what it sees

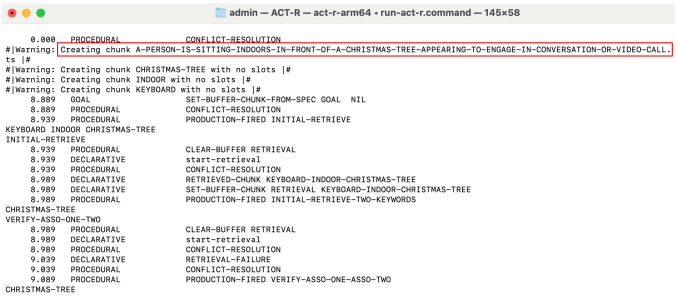


Fig. 4. ACT-R model creates chunks for keywords and phrase

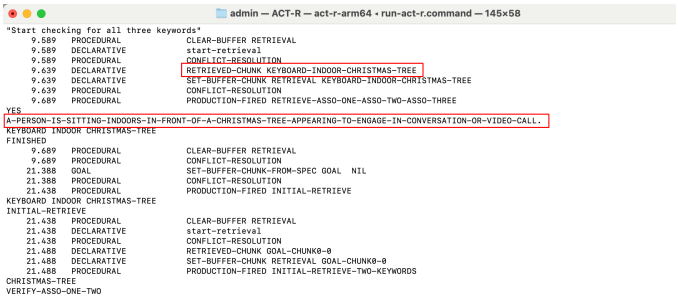


Fig. 5. ACT-R model retrieves chunk with keywords and phrase from declarative memory

keywords, into the declarative memory in order to retrieve them later in the outlined manner.

D. Prompting the LLM

We chose GPT-4o to create the conversational parts of the robot. The OpenAI API provides various hyperparameters that can be used to control the model behavior during an API call. We set the values of *temperature*, *presence penalty* and *frequency penalty* to zero in order to obtain consistent responses and exclude any randomness as far as possible. For the instruction of the GPT model, we used prompts with zero-shot prompting [4] for the system role to have the LLM perform the desired tasks as a completion task.

For chunk generation, including a phrase that is to be stored in the *'phrase'* slot describing the image sent to GPT-4o, we have provided a system prompt like the following *'I am robot, my name is Pepper. I don't answer but create three keywords followed by a sentence with a short description describing what is in this image. The three keywords are enclosed by square brackets. The short sentence is enclosed by round brackets, for example [person, indoor, computer] (A sentence with a short description describing what is in this image).'*. The brackets only serve to facilitate subsequent processing and separation of different contents in the application.

For augmenting the prompt with a retrieved recollection we used for example *'I am robot, my name is Pepper. I can see what is described in the following:'* followed by the *phrase* recalled from declarative memory. In this way, the LLM could receive and process information about visual recollections stored in the past if similar sensory impressions and thus similar keywords are currently present.

IV. Results

Figure 4 shows the generation of individual chunks from the keywords *'keyboard'*, *'indoor'*, *'christmas tree'* and in a frame the phrase *'A person is sitting indoors in front of a christmas tree appearing to engage in conversation or video call.'* supplied by the LLM in the running ACT-R model. These chunks were automatically transferred to the declarative memory of the cognitive model, not only separately, but also as a combination of keywords and phrase. To be able to save keywords and phrases as chunks, spaces had to be replaced by a '-' due to ACT-R limitations. The procedure was reversed when the phrase was used for prompt augmentation.

A retrieval of a recollection by the ACT-R model is shown in Figure 5 in framed areas. The model searches the stored memory chunks using the supplied keywords to find a matching recollection, the content of which is passed from the *'phrase'* slot to the robot application for use in prompt augmentation. The robot then included the knowledge from declarative memory in its answer – especially when asked directly about visual impressions.

V. Discussion

On the one hand, cognitive models can be used within the framework of their cognitive architectures to anticipate human behavior and thus make it more comprehensible. On the other hand, cognitive processes can use declarative and procedural memory to produce decisions which, when used to control the actions of a robot, allow the robot to act in a more humane way. Such a system can therefore serve both to make human actions intelligible for the machine in HRI and to make the robot's actions more human and therefore easier

for people to understand. The possibility of using cognitive architectures to unlock human-like judgment and decision-making capabilities such as instance-based learning and intuitive decision making for social robots can be an opportunity for greater acceptance and trust through more common ground in the way we think. With the help of LLMs and VLMs and the corresponding sensors, robots can access and interpret the same information that is available to humans. The use of declarative and procedural memory in cognitive models enables the data provided by robot sensors and language models to be processed in a human-centered way. We only outline the possibilities here with our example application for storing visual impressions together with keywords and using them for prompt augmentation. In general, there are far more conceivable use cases and options available for a programmatic implementation of cognitive models with ACT-R in combination with social robots.

VI. Limitations

The proposed system and procedure is not without shortcomings and open problems. It is well known that LLMs and VLMs are sometimes prone to hallucinations. One possible way to reduce the risk of such hallucinations and the reproduction of made-up statements by an LLM such as OpenAI's GPT would be to provide the language model with additional information tailored to individual scenarios via the system prompt. To concretize the prompt information, relevant memories of a cognitive system could be a useful addition. However, even such a method of constraining the LLM does not offer absolute certainty for the exclusion of hallucinations.

So far, we have only tested our method with a few memory contents or recollections, but we consider it scalable with regard to more complex cognitive models and the use of concepts such as *forgetting*, *learning new facts* and *reinforcement of recollections* or further possibilities of the cognitive architecture of ACT-R.

The time required to retrieve relevant recollections could play a limiting role depending on the amount of memory chunks available. In addition, for each interaction with the human user, our system requires two consecutive API calls to the GPT model, both of which have a certain latency. Together, these time aspects may cause a noticeable and unnatural delay in the interaction. And it also seems to be a disadvantage and increases the complexity that the implementation of the cognitive architecture runs as a standalone version on an extra computer and not on the robot itself. This problem concerns the Pepper robot and could possibly be solved by using other robot models and thus other ways of implementing the cognitive architecture.

VII. Conclusion and future work

We proposed a design and development approach for a combined system consisting of an ACT-R cognitive model and a humanoid social robot. Recollections from the declarative memory of the ACT-R model could be retrieved using real-world data obtained by the robot via an LLM with vision abilities. The procedural memory, which consists of the productions of the cognitive model, was used to retrieve these recollections and return them to the robot as instructions for action. In addition, real-world data captured by the robot could be stored as memory chunks in the cognitive model's declarative memory.

We improved the correctness and accuracy of GPT-4o's reasoning capabilities by using recollections for prompt augmentation. Our proof-of-concept application delivered keyword labeled visual impressions of the robot to the declarative memory or retrieved recollections based on these impressions. This method can of course also be used with other than visual data. We testet this system with the social robot Pepper. However, this method can also be used with other robot models and also independently of HRI scenarios.

Regarding explainability aspects, our system provides approaches for a possible constraint of LLMs to generate robot utterances in HRI by comprehensible memory contents. The type of possible connection or integration of the cognitive model depends on the robot's operating system and whether or which ACT-R implementations are available for it. For example, there is a direct implementation for Python, which means that the standalone version of ACT-R on an external computer and thus also the TCP/IP connection between the cognitive model and the robot application could be avoided.

The use of a cognitive architecture such as ACT-R enables the inclusion and investigation of further cognitive principles and processes in interaction with a social robot and LLMs independent of declarative memory retrieval. Future work should consider these aspects in more detail. Further experiments are needed to optimize the ACT-R models and system prompts for the LLM, as well as ongoing evaluation in studies with different people and different robot systems for various tasks. We are confident that this will open up a wide range of possibilities for future research into how cognitive architectures and their models can add a human touch to a social robot in HRI.

References

[1] Aldebaran, United Robotics Group and Softbank Robotics. Pepper. Technical report, 2025. URL https://www.aldebaran.com/en/pepper.

[2] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. 111(4):1036–1060, 2004. doi: 10.1037/0033-295X.111.4.1036.

[3] Manuel Birlo and Adriana Tapus. The crucial role of robot self-awareness in hri. In *Proceedings of the 6th International Conference on Human Robot Interaction*, pages 115–116, 03 2011. doi: 10.1145/1957656.1957688.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel

Ziegler, Jeffrey Wu, Clemens Winter, and Dario Amodei. Language models are few-shot learners, 05 2020.

[5] Carnegie Mellon University. Act-r software. Technical report, 2024. URL http://act-r.psy.cmu.edu/software/.

[6] Souvik Dubey, Ritwik Ghosh, Mahua Dubey, Subhankar Chatterjee, Shambaditya Das, and Julián Benito-León. Redefining cognitive domains in the era of chatgpt: A comprehensive analysis of artificial intelligence's influence and future implications. *Medical Research Archives*, 12, 06 2024. doi: 10.18103/mra.v12i6.5383.

[7] Akash Ghosh, Arkadeep Acharya, Sriparna Saha, Vinija Jain, and Aman Chadha. Exploring the frontier of vision-language models: A survey of current methodologies and future directions, 2024. URL https://arxiv.org/abs/2404.07214.

[8] Cleotilde Gonzalez, Javier F Lerch, and Christian Lebiere. Instance-based learning in dynamic decision making. *Cognitive Science*, 27(4):591–635, 2003. ISSN 0364-0213. doi: https://doi.org/10.1016/S0364-0213(03)00031-4. URL https://www.sciencedirect.com/science/article/pii/S0364021303000314.

[9] Cleotilde Gonzalez, Varun Dutt, and Christian Lebiere. Validating instance-based learning mechanisms outside of act-r. *Journal of Computational Science*, 4(4):262–268, 2013. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2011.12.001. URL https://www.sciencedirect.com/science/article/pii/S1877750311001086. PEDISWESA 2011 and Sc. computing for Cog. Sciences.

[10] Miguel Á. González-Santamarta, Francisco J. Rodríguez-Lera, Ángel Manuel Guerrero-Higueras, and Vicente Matellán-Olivera. Integration of large language models within cognitive architectures for autonomous robots, 2024. URL https://arxiv.org/abs/2309.14945.

[11] Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning, 2023. URL https://arxiv.org/abs/2311.17842.

[12] Hyeongyo Jeong, Haechan Lee, Changwon Kim, and Sungtae Shin. A survey of robot intelligence with large language models. *Applied Sciences*, 14(19), 2024. ISSN 2076-3417. doi: 10.3390/app14198868. URL https://www.mdpi.com/2076-3417/14/19/8868.

[13] Daniel Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, 2011. ISBN 9780374275631 0374275637.

[14] Kento Kawaharazuka, Yoshiki Obinata, Naoaki Kanazawa, Kei Okada, and Masayuki Inaba. Robotic applications of pre-trained vision-language models to various recognition behaviors. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, page 1–8. IEEE, December 2023. doi: 10.1109/humanoids57100.2023.10375211. URL http://dx.doi.org/10.1109/Humanoids57100.2023.10375211.

[15] Kobe Knowles, Michael Witbrock, Gillian Dobbie, and Vithya Yogarajan. A proposal for a language model based cognitive architecture. *Proceedings of the AAAI Symposium Series*, 2024. URL https://api.semanticscholar.org/CorpusID:267206594.

[16] Daniel Leidner. Toward robotic metacognition: Redefining self-awareness in an era of vision-language models. *40th Anniversary of the IEEE Conference on Robotics and Automation*, 2024. URL https://elib.dlr.de/207493/.

[17] Evelina Leivada, Gary Marcus, Fritz Günther, and Elliot Murphy. A sentence is worth a thousand pictures: Can large language models understand hum4n l4ngu4ge and the w0rld behind w0rds?, 2024. URL https://arxiv.org/abs/2308.00109.

[18] Artem Lykov, Mikhail Litvinov, Mikhail Konenkov, Rinat Prochii, Nikita Burtsev, Ali Alridha Abdulkarim, Artem Bazhenov, Vladimir Berman, and Dzmitry Tsetserukou. Cognitivedog: Large multimodal model based system to translate vision and language into action of quadruped robot. HRI '24, page 712–716, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703232. doi: 10.1145/3610978.3641080.

[19] Qian Niu, Junyu Liu, Ziqian Bi, Pohsun Feng, Benji Peng, Keyu Chen, Ming Li, Lawrence KQ Yan, Yichao Zhang, Caitlyn Heqi Yin, Cheng Fei, Tianyang Wang, Yunze Wang, and Silin Chen. Large language models and cognitive science: A comprehensive review of similarities, differences, and challenges, 2024. URL https://arxiv.org/abs/2409.02387.

[20] OpenAI. The most powerful platform for building ai products. Technical report, 2025. URL https://openai.com/api/.

[21] Thomas Sievers and Nele Russwinkel. How to use a cognitive architecture for a dynamic person model with a social robot in human collaboration. *CEUR Workshop Proceedings*, 2024. URL https://ceur-ws.org/Vol-3794/paper2.pdf.

[22] Thomas Sievers, Nele Russwinkel, and Ralf Möller. What am I? – complementing a robot's task-solving capabilities with a mental model using a cognitive architecture. *CEUR Workshop Proceedings*, 2024. URL https://ceur-ws.org/Vol-3906/paper1_BAILAR.pdf.

[23] Thomas Sievers, Nele Russwinkel, and Ralf Möller. Grounding a social robot's understanding of words with associations in a cognitive architecture. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, pages 406–410. INSTICC, SciTePress, 2025. ISBN 978-989-758-737-5. doi: 10.5220/0013144200003890.

[24] Ron Sun. Can a cognitive architecture fundamentally enhance llms? or vice versa?, 2024. URL https://arxiv.org/abs/2401.10444.

[25] Ryan Tan, Harry Yang, Sean Jiao, Liuyang Shan, Chenxi Tao, and Roger Jiao. Smart robot manipulation using gpt-4o vision. In *7th European Industrial Engineering and Operations Management Conference, Augsburg, Germany*. IEOM Society International, 2024. doi: 10.46254/EU07.20240272.

[26] Robert Thomson, Christian Lebiere, John R. Anderson,

and James Staszewski. A general instance-based learning framework for studying intuitive decision-making in a cognitive architecture. *Journal of Applied Research in Memory and Cognition*, 4(3):180–190, 2015. ISSN 2211-3681. doi: https://doi.org/10.1016/j.jarmac.2014.06.002. URL https://www.sciencedirect.com/science/article/pii/S2211368114000539. Modeling and Aiding Intuition in Organizational Decision Making.

[27] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Gpt-4v(ision) for robotics: Multimodal task planning from human demonstration, 2024. URL https://arxiv.org/abs/2311.12015.

[28] Jiaqi Wang, Enze Shi, Huawen Hu, Chong Ma, Yiheng Liu, Xuhui Wang, Yincheng Yao, Xuan Liu, Bao Ge, and Shu Zhang. Large language models for robotics: Opportunities, challenges, and perspectives. *Journal of Automation and Intelligence*, 2024. ISSN 2949-8554. doi: https://doi.org/10.1016/j.jai.2024.12.003. URL https://www.sciencedirect.com/science/article/pii/S2949855424000613.

[29] A. Werk, S. Scholz, T. Sievers, and N. Russwinkel. How to provide a dynamic cognitive person model of a human collaboration partner to a pepper robot. *ICCM*, 2024. URL https://mathpsych.org/presentation/1452.

[30] Siyu Wu, Alessandro Oltramari, Jonathan Francis, C. Lee Giles, and Frank E. Ritter. Cognitive llms: Towards integrating cognitive architectures and large language models for manufacturing decision-making, 2024. URL https://arxiv.org/abs/2408.09176.

[31] Takahide Yoshida, Suzune Baba, Atsushi Masumori, and Takashi Ikegami. Minimal self in humanoid robot "alter3" driven by large language model, 2024. URL https://arxiv.org/abs/2406.11420.

Proceedings of the 23rd International Conference on Cognitive Modelling (ICCM 2025)

Empirical Test of a Formal Framework of Forgetting

Sara Todorovikj (sara.todorovikj@hsw.tu-chemnitz.de)
Daniel Brand (daniel.brand@hsw.tu-chemnitz.de)
Marco Ragni (marco.ragni@hsw.tu-chemnitz.de)
Predictive Analytics, Chemnitz University of Technology
Straße der Nationen 62, 09111 Chemnitz, Germany

Abstract

Forgetting is a fundamental cognitive process that allows to efficiently discard outdated and irrelevant knowledge and manage information for humans and artificial systems alike. Originating from the latter, Beierle et al. (2019) propose a formal framework for modeling forgetting operators from a commonsense point of view. In this article, we aim for bridging the gap by investigating the formal forgetting operators with respect to their effects on human cognition utilizing an abstract experimental paradigm – the Counting Game. Using participants' accuracy and response time in rule-based tasks, we examine if and how the forgetting operators affect human performance in response to rule manipulations. We found that the addition, contraction and revision of a rule have the most remarkable effect on performance. Therefore, we modeled them using features describing the rule system and current scenario to predict the change in accuracy they prompt. By analyzing the effect that the operators have on humans when used to manipulate rule systems, we make a step towards further modeling the formal aspect of changes to rule systems in applications in order to understand the potential effect they have on the users.

Keywords: Forgetting Operations; Empirical Validation; Rule Updating

Introduction

Humans are constantly exposed to new situations that require them to use various cognitive mechanisms to categorize and interpret information. Understanding how we acquire and update such categorization rules is central in cognitive science. Rule changes, in general, play a crucial role in problem solving, decision making and learning. However, sometimes knowledge is irrelevant or outdated and requires to be discarded. As an everyday phenomenon, forgetting is a fundamental cognitive process that allows humans to efficiently manage and restructure information. Beyond psychology, forgetting has been studied in different contexts, like logical systems, e.g., propositional or first order logic (Delgrande, 2014), or belief revision (Alchourrón, Gärdenfors, & Makinson, 1985). However, a generally accepted theory or formalization that unifies all such approaches does not yet exist. Towards that goal, Beierle, Kern-Isberner, Sauerwald, Bock, and Ragni (2019) propose a framework that conceptualizes different forms of forgetting, offering a structured approach to modeling belief change and formalizing forgetting from a common-sense point of view.

Experimental approaches in the field of rule changes often use rule-based systems to analyze how individuals update concepts. For example, Brand, Dames, Puricelli, and Ragni (2022) investigated the cognitive costs of adding new categorization rules or altering previously learned ones, finding that modifying an existing rule was less cognitively demanding than adding a new one, indicating the presence of processes depending on the number of rules. Additionally, they found that performance improves when new rules align with old ones, but in case of conflicting situations, this was no longer apparent. At their core, rules function as if-then conditions: if *a precondition is satisfied* then *a corresponding action follows*. This structure underlies processes such as learning to categorize (Maddox, Ashby, Ing, & Pickering, 2004; Maddox, Ashby, & Bohil, 2003), implementing intentions (Oettingen & Gollwitzer, 2010; Gollwitzer & Brandstätter, 1997) and is central to cognitive architectures like ACT-R (Anderson, 2007), among others. Studying how rules are acquired, updated and modified is essential for understanding how individuals adapt to changing environments.

To bridge the gap between formal models and human cognition, we utilize a new experimental paradigm designed to empirically test the predictions made by the formal framework. Using controlled rule-based tasks, we examine how and to which extent participants adapt and apply their knowledge in response to rule changes. This paradigm allows us to investigate different forgetting operators through rule manipulations and assess associated cognitive costs. Tying formally defined forgetting operators that describe changes in knowledge and rule systems can also allow their application in human-centered fields. If the effects (e.g., their impact on cognitive load) that the operators have when used by humans can be determined and understood, those operators can be used as a means to model not only the formal aspect of changes to rule systems in applications (e.g., software systems and work processes), but also provide estimates on the effects such changes will have on the users.

In this article, we aim to provide an empirical validation of formal definitions of forgetting operators. More specifically, we examine whether human behavior aligns with forgetting patterns predicted by the formal framework, by looking into how these operators influence accuracy and response time. Building up on our analysis results, we present linear regression models for three rule operators that aim to represent the effect they have on accuracy by considering features that describe the participants' knowledge base and the impact a rule has on the specific scenario.

Theoretical Background

Within the framework for modeling forgetting operators presented by Beierle et al. (2019), given a belief state that can have any type of representation, such as logical formulas or rankings over possible worlds, an agent's inference relation determines what follows from the beliefs. Using ordinal conditional functions (OCFs) they can assign degrees of plausibility, allowing for more nuanced reasoning and belief modification and removal, i.e. forgetting. We focus on the following five forgetting operators.

Contraction is an operator describing explicit intention to remove information, e.g. a navigation system being informed about a permanent closure of a street. Formally, contraction of a belief results in not believing the proposition afterwards, leading to a consistent belief state.

Revision involves presentation of new information that contradicts already established knowledge, prompting the forgetting of the old information. E.g., receiving new information that a person who was previously a student is now a staff member. In the case of revision, forgetting is implicit - given new information, any beliefs conflicting with it have to be forgotten in order to adopt the new, prioritized belief. It can be made explicit formally, by first contracting the old information and then adding the new one.

Abstraction is a common change operation that involves generalizing specific beliefs or rules in order to create a more broadly applicable one. For example, a rule can be that an object with two wheels and a bike bell is a bicycle, whereas another rule can state that an object with two wheels and no bike bell is also a bicycle. From these rules, a simplified rule can be abstracted, namely that an object with two wheels is a bicycle. Both, deductive and inductive reasoning may be involved in deriving a more general principle, which in turn allows to potentially forget the more complicated initial rules.

Conditionalization is an operator that restricts a belief set by omitting ones that do not adhere to a specific case, context or precondition. Inspired by probabilistic conditionalization where posterior beliefs are determined by conditional probabilities, the operator centers the beliefs around the condition leading to forgetting when it is not satisfied.

Fading out is a gradual forgetting process where information becomes increasingly difficult to retrieve over time, especially in the case of infrequent use. For example, a PIN code for a credit card that is rarely used is more likely to be forgotten eventually. Formally, it describes the increase in required effort to make an inference based on a set of beliefs in a later point in time.

Commonly, such belief systems are illustrated with declarative instead of procedural knowledge (e.g., like in the PIN code example above) but are in fact applicable on the latter as well. This allows to utilize the presented operators in the context of rule changes and rule learning, providing a formal framework for describing changes to a rule system. Thereby, the question arises if and how those formal forgetting operators are performed in human memory, and what the cognitive costs of each operation are. To this end, first investigations of *revision* have been performed, indicating that updating and altering an existing rule is less cognitively demanding than learning a new rule (Brand et al., 2022). However, in order to assess the applicability of the forgetting operators also as a cognitive modeling framework, all operators need to be investigated under controlled and comparable conditions. To this achieve this, we utilize a new abstract experimental paradigm based on shapes and colors, called the *Counting Game* (anonymized). Its goal is to provide a mean to empirically investigate and compare cognitive costs of forgetting operators, among many other concept changes.

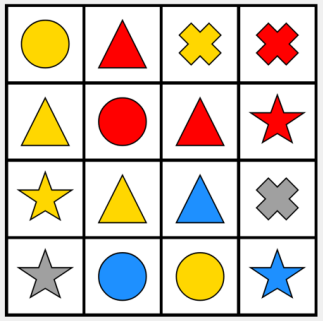


Figure 1: Representation of the Counting Game showing a trial whose winner is yellow. A rule "Red triangles count twice" changes the winner to red.

Method

The Counting Game consists of *trials*, which are 4×4 grids of 16 objects with various *shape* (circle, triangle, cross, star) and *color* (red, yellow, blue, gray), as shown in Fig. 1. Each trial has a winner – the color that has the largest amount of objects, among the "competing" colors – red, yellow and blue. A valid trial has only one winning color (which cannot be gray) and at most half of the total number of elements can be of the same color. A study implemented within the Counting Game paradigm presents multiple trials to participants and prompts them to determine which color is the winner as fast as possible. To motivate fast, but also accurate answers, we introduce a scoring system that awards 10 points for each correct answer, with an additional bonus timer counting down from 20 seconds and awarding 1 point per second remaining. So, a correct answer with 8 seconds left on the timer awards 10 + 8 points). Incorrect answers do not award points, regardless of how fast they were given.

At the beginning, participants simply count objects – each one, regardless of color and shape, counts as one. Throughout the game, we introduce two types of rules that change how specific objects are counted:

1. *Count-twice* rules state that objects of a certain color and shape count twice from now on (e.g., "Red stars count twice.")

2. *Count-as* rules state that objects of a certain color and shape count as a different color from now on (e.g., "Blue

Table 1: Rules implementing the tests of the forgetting operators.

Operator	Rules
Contraction	Blue stars count twice. Blue stars count as one again.
Revision	Red circles count as yellow. Red circles count as blue.
Abstraction	Red circles count as yellow. Red stars count as yellow. Red crosses count as yellow. Red triangles count as yellow.
Conditionalization	Gray crosses count as blue.
Fading Out	Blue crosses count twice.

crosses count as yellow.")

Using these rules, we test the previously introduced forgetting operators, as shown in Table 1. We manipulate the individual's rule system within the game by changing how specific objects count, which then in turn allows to quantify the impact of the forgetting operations on performance.

In this work, a *Phase* refers to a stage within a round when a specific rule configuration is active. All rounds start with no rules in *Phase 0* and progress through multiple phases. A new phase begins each time a rule is introduced, updated or removed. So, *Phase 1* begins after the first rule, *Phase 2* after the second, and so on. Each phase consists of the same trials in randomized order. That allows us to investigate how participants adapt to new rules and how/if prior rules affect the application of recent ones. All rules have corresponding *critical trials*. They are trials whose winner changes after a rule is applied. For example, the trial depicted in Fig. 1 is critical for the rule "Red triangles count twice" as its application changes the winner from yellow to red.

Forgetting Operators

An overview of rules used to implement the forgetting operators is provided in Table 1. Contraction is implemented by providing one *count-twice* rule and explicitly withdrawing it later. In the case of revision, we give two *count-as* rules, such that the second one overrides the first one, implicitly making it irrelevant. For abstraction, the participants got four *count-as* rules that cover all shapes of a given color, prompting them to ultimately replace the four rules with one, e.g. "Red shapes count as yellow". The presence of gray elements in a trial is optional, therefore for conditionalization we use specific *count-as* rules where the initial color is gray, introducing the precondition that *if* there are gray shapes, they will count as a different color. In the case of fading out, we only use one *count-twice* rule, however, here the temporal component of the operator is important. We had multiple fading out rounds manipulating the amount of time (trials) between the rule presentation and the corresponding critical trials. We added filler rules, so that the rounds are comparable with four rules each.

Participants

Each participant was randomly and evenly assigned to one of the five operator conditions. To prevent unwanted effects or bias, each participant had a randomly assigned set of colors and shapes. So, a rule "$color_1$ $shape_1$ count twice" could be about red crosses for one person and yellow stars for another. The general structure of the rounds, rules and trials is otherwise preserved.

We obtained data from 117 participants recruited on Prolific[1]. The experiment was performed online as a web-experiment. Participants needed on average 25 min. to complete the experiment and received monetary compensation at the end. All participants were native English speakers. The participants first had to provide consent to have their experimental data stored. Afterwards, they were introduced to the game and its rules, the possible shapes and colors, the goal to find the winner as fast as possible and the scoring system. Then, they did a practice round with a few tasks in order to get accustomed to the format. Once finished with the instructions, the participants started with the study. The data, analysis and modeling scripts developed for this article are available on GitHub[2].

Analysis

We identified as outliers four participants whose mean accuracy score diverged by at least 2 standard deviations away from the mean and excluded them from the analysis. The final number of participants is 113. The overall accuracy across all participants and conditions was 83.62% (Median $= 100\%, SD = 37.01\%$). The mean accuracy of trials critical for at least one rule was 74.62% and 95.68% for non-critical trials. The median response time (RT) across all trials was 2.06s. Critical trials had a higher median RT (2.22s) in contrast to non-critical trials (1.90s).

Contraction We examine the average response accuracy of critical and non-critical trials when no rules were present, after a rule addition and after that rule's contraction, as presented in Table 2. The accuracy decreases once a rule is added (Wilcoxon: $z = 68.50$, $p = .010$) and we identified that all errors are due to participants not applying the rule. Once the rule is contracted, the accuracy improves once again ($z = 4.00$, $p = .014$). Analyzing the errors, we discovered out that 93.75% of wrong responses were because the rule was still applied, i.e. it was not contracted. Furthermore, we confirmed that correct responses after contraction were not a consequence of not applying the rule in the first place (Spearman $\rho = 0.13$, $p = .227$). In general, contraction should fa-

[1] https://www.prolific.com

[2] https://github.com/saratdr/ICCM2025-CountingGame

Table 2: Average response accuracy on critical and non-critical trials in the contraction and revision round. Crit and NonCrit indicate critical and non-critical, respectively.

		No rule	Rule	After Op.
Contraction	Crit	88.89%	63.33%	82.22%
	NonCrit	91.11%	94.44%	91.67%
Revision	Crit	85.71%	75.19%	80.45%
	NonCrit	94.30%	92.98%	94.74%

cilitate rule application, reducing the cognitive load by explicitly removing a rule. To investigate this effect, we compared two scenarios where three rules were added. We focus on response accuracy of critical trials for the last rule if the second rule was previously contracted and not (Mean: $contr = 76.79\%$, $notcontr = 57.81\%$). Since these scenarios used different trials and rules, we normalized the values by the participants' general performance on them in a phase with no rules. We observed a marginal effect suggesting that contraction may indeed ease the process of remembering and applying rules (Mean, normalized: $contr = 87.61\%$, $notcontr = 72.87\%$; Mann-Whitney: $U = 149.50$, $p = .062$).

Revision For revision, the accuracies are depicted in Table 2. Like in the case of contraction, non-critical trials remained unaffected and stayed at a consistently high accuracy of over 90%. For critical trials, a similar pattern to the contraction emerged, with a dip in performance when the rule was added, with a slight improvement after revision. In this case, however, participants performed better after the addition of the rule, which was likely due to an easier structure of the critical trials compared to those of the contraction condition. However, since revision does only change the rule to a new rule with an arguably comparable difficulty, it seems unexpected that performance would increase compared to the simple addition of a rule. A possible explanation could be that participants, who forgot or did not learn the first rule properly, got essentially a "second chance", since the revision removed the rule in question. 75.76% of the errors after a rule was added (with a total of 24.81% of the critical trials being answered incorrectly) were in fact due to the rule not being applied, indicating that the rule was not perfectly learned. This at least partially supports the possible explanation, that participants indeed simply did not learn the first rule properly, but were better prepared later. After the rule was revised, 7.69% of the remaining errors (19.55%) was due to the previously learned rule still being present. This indicates that the revision itself was performed successfully, leaving little artifacts of the old rule.

Abstraction In the case of abstraction, we analyzed how the response accuracy on critical trial changes after the addition of each rule. Thereby, the mean accuracy decreased after each rule until the final rule necessary for performing abstraction was introduced (70.83%, 56.25%, 43.06% to 47.22%, for the respective phases). We can immediately notice a tendency of accuracy to decline with each rule addition, except for a marginal increase after the last rule. Given that the addition of the last rule enabled participants to abstract over all four rules and apply only one, a better performance is indeed expected in the last phase. Such increase was found for the majority of the participants (58.33%).

Table 3: Response accuracy ratio between Phase 1 (rule applied) and Phase 0 (no rules) on irrelevant trials that do not contain the object affected by the rule for conditionalization and the rest, and trials with no gray elements for conditionalization.

Rule	Trials	Accuracy Ratio
Conditionalization	Irrelevant	99.78
	No Gray	104.58
Rest	Irrelevant	101.58

Conditionalization For conditionalization, an important distinction is between the effect of the knowledge level (i.e., are rules not present/active at the moment, so that they don't affect cognitive load) and the effect of rule application itself (i.e., the counting itself is easier when no additional rules need to be considered). Thereby, the important part for our evaluation is the former, since it is the one that the logical framework is concerned with. However, in practice, both effects occur together, making it hard to distinguish between them. To this end, we first investigate the effects on *irrelevant* trials, i.e., trials that do not contain elements affected by the rule. That way, we can see if the rule is generally less present when it is not applicable compared to other rules. Additionally, since gray is a color only relevant due to the conditionalization rule, we also consider trials with no gray elements at all. Table 3 shows the ratio between the accuracy of *Phase 1* (where the first rule is present) and *Phase 0* (where no rule is present) for the conditionalization on irrelevant trials and trials without gray shapes, as well as for the other conditions as a comparison. It becomes apparent that the accuracy does not substantially change for either condition, and is also only slightly improved when gray shapes are not present at all, indicating that conditionalization does seem to affect the cognitive load mostly on the applicational level.

To estimate the effect on the applicational level, we compare the response times between relevant and irrelevant trials, as well as trials with and without any gray elements. Thereby, no significant effect was shown between relevant and irrelevant trials (Median: $rel = 2.65s$, $irrel = 2.66s$, Wilcoxon: $z = 368.5$, $p = .061$). For trials with and without gray elements, however, the effect on the response time was significant Median: $rel = 2.76s$, $irrel = 2.51s$, Wilcoxon: $z = 389.5$, $p = .026$). Put together, this shows that the conditionalization of the prominent surface feature (i.e., gray color)

Table 4: Average response accuracy on critical trials immediately after rule addition and at the end of the round. Rounds 1, 2 and 3 have an increasing number of trials. Round 4 introduces additional rules, irrelevant to the fading out rule.

Round	After rule	End of round	Mean Difference
1	69.44%	51.39%	20.32%
2	66.67%	48.61%	20.53%
3	76.39%	45.83%	32.50%
4	65.28%	20.84%	44.65%

did help to accelerate the process of solving the tasks, however, as soon as the more detailed condition (i.e., color and shape) comes into play, that advantage vanished.

Fading Out In the case of fading out, we tested participants' performance in four different rounds. In the first three rounds, participants were presented with only one rule in *Phase 1* followed only by trials and no additional rules. The length of the rounds based on the amount of presented trials corresponds to 3, 4 and 5 phases, respectively. The critical trials were presented once among the first 12 trials after the rule and once among the last 12 trials of the round. That way, we ensured an infrequent application of the rule and we can examine the temporal effect of the operator. The average response accuracy and their difference between the first and last trial presentation are presented in Table 4. While a decrease in accuracy over time can be noted, it is not a drastic one, with a 32.50% difference in the longest round.

In the fourth round, we used the same principle of only presenting the critical trials of the relevant rule at the beginning and end, but we also added rules at the beginning of each phase. This led to a much substantial negative effect on performance, suggesting that information is rather increasingly difficult to retrieve over manipulations than time. In order to further confirm the effect of additional rules within this round, we looked into their corresponding critical trials. In Table 5 we can see that a rule addition leads to a lower accuracy for its critical trials. Moreover, introducing yet another operation causes a further substantial decline for trials critical for previous rules, indicating that a recency effect potentially comes into play as well.

Modeling

In our analyses, we found strong evidence for the presence and effect of three operators involved in rule manipulation: *addition* (implicitly used to build up to rule system), *contraction* and *revision*. In order to further evaluate their impact and estimate the associated cognitive costs we use linear regression models to predict the difference in accuracy that an applied operator induces on a trial per trial basis. To account for the influence of the cognitive demand of the participants' current rule system, a feature representing the amount of rules that are currently active in a given trial was included (*CurrActive*). In order to contrast the abstract nature of the operators with their dependency on the context in which they are applied, we add *RelevantBool* and *RelevantNum* as model features. They incorporate the current scenario by indicating if and how many relevant shapes for the currently added, contracted or revised rule are present in the trial, regardless if it's a critical trial one or not.

Table 5: Additional rules in Round 4 of fading out. The second rule (R2) is introduced in Phase 2, the third (R3) in Phase 3. R2 is contracted in Phase 4. R2 Crit and R3 Crit are critical trials for the second and third rule, respectively.

		Phase 2	Phase 3	Phase 4
Accuracy	R2 Crit	58.34%	25.00%	80.56%
	R3 Crit	80.56%	72.22%	36.11%

In Table 6 we present the means and standard deviations of the accuracy differences that the models aim to fit, along with the models' mean absolute errors (MAE) and parameter values. For the critical trials the *RelevantBool* parameter is irrelevant, since all of them contain relevant shapes by default. Additionally, the parameter *CurrActive* is irrelevant for revision, since in our experiment, it always took place in the same phase to facilitate the statistical analyses. We first examine our target variables and model fits. Addition and revision of a rule harm performance in critical trials, whereas contraction improves it. This is in line with the intuition behind the operators. In contrast, the operators seem to have minimal effect on average for non-critical trials. In terms of the impact on performance on critical tries, revision is substantially less impactful compared to addition, which is in line with the results of Brand et al. (2022). Thereby, revision is showing a performance similar to a combination of contraction and addition.

Considering the model fits, it becomes apparent that revision achieved a substantially better fit, while both addition and contraction have R^2 values close to 0, thereby not being able to utilize the given features. This, however, is likely due to the fact that revision is only present in isolated cases (i.e., conditions that explicitly test revision), while especially addition is present across all conditions, making the data more noisy and less dependent on the limited features. Nevertheless, the coefficients provide some insight about the used features. Compared to the number of relevant shapes, the number of active rules has a substantially lower importance. This highlights the strong relationship between the context (i.e., the actual composition of shapes in a trial) and the rules, while the effects of the manipulations to the rule system itself are rather indirect.

Discussion

In this article, we investigated a formal framework of forgetting operators introduced by Beierle et al. (2019) on a cognitive level, in particular for describing changes of a rule system. Therefore, we utilized a new experimental paradigm to

Table 6: Mean and standard deviation (STD) of differences in accuracy after adding, contracting or revising a rule for critical and non-critical trials. Mean absolute error (MAE), coefficient of determination (R^2) and parameter values of our models fit. *CurrActive* is the amount of currently applicable rules, *RelevantBool* indicates the presence of shapes affected by the rule in the trial and *RelevantNum* is the amount of those relevant shapes. The parameter *RelevantBool* is irrelevant for critical trials, since all of them contain relevant shapes. *CurrActive* is irrelevant for revision, since it always took place during the same phase.

Trials	Operator	Mean	STD	MAE	R^2	Model Parameters			
						CurrActive	*RelevantBool*	*RelevantNum*	Intercept
Critical	Addition	−19.04	17.49	13.00	0.11	−0.003	–	0.313	−0.356
	Contraction	13.87	20.64	15.04	0.09	0.089	–	0.114	−0.039
	Revision	−10.00	12.99	6.14	0.62	–	–	0.477	−0.383
Non-critical	Addition	−2.40	16.71	10.85	0.07	−0.019	−0.002	0.302	−0.033
	Contraction	−4.42	17.20	11.33	0.10	−0.024	0.007	−0.336	0.038
	Revision	5.00	9.52	4.04	0.46	–	−0.185	0.496	0.046

conduct a study in which participants had to apply various rules in a simple counting task: Participants had to decide which color was the most prominent in a set of shapes. Within this paradigm, distinct manipulations of the counting rules (e.g., some shapes are ought to be counted twice) allowed to assess the impact the forgetting operators have on performance. We conducted a study within the paradigm in which we used rule manipulations in order to assess the effects of forgetting operators and evaluate the predictions made by the formal framework. This allowed us to gain insights about how and if those formal operators are affecting human rule manipulation in procedural memory.

Thereby, we investigated five operations: contraction, revision, abstraction, conditionalization, and the fading out of a rule if it is not utilized over time. Implicitly, we also tested the addition of a new rule, as the only not forgetting-related operator. We found that adding a rule significantly decreased the accuracy of solving the task, and even to a bigger extend as the revision of a rule – which is in line with the findings of Brand et al. (2022). For contraction, i.e., the removal of a rule, we found that performance was restored to almost it's original level. Furthermore, errors after contraction mostly related to the rule still being applied, which indicates that artifacts of the rule still remain. Additionally, we found typical effects of directed forgetting (e.g., Dames & Oberauer, 2022; Dames, Brand, & Ragni, 2022): being able to forget a rule, allowed participants to free resources for learning other rules. This also can explain the lower impact on performance of a revision compared to a simple addition, since revision can be interpreted as the combination of forgetting of an old rule and learning of a new one.

Furthermore, we found evidence that abstraction, i.e., the simplification of several specific rules into a more general one, seemed to take place. While the addition of rules typically decreases performance, the majority of participants showed improvements after the last missing rule was added to allow for abstraction. For Conditionalization, we found that the effects are mostly related to the application of rules itself, and do not influence the rule system itself. While participants improved when a rule was not relevant for a trial, the type of the rule itself did not affect performance on the trials where it was not applicable, thereby giving no indication for a difference in general cognitive load.

Finally, for the last operator, fading out, we found a subtle trend that longer phases where a rule is not utilized indeed reduce performance when it is relevant again. However, the effect is only within the magnitude of error, thereby not warranting a conclusion. Overall, this can also be an effect of the generally short time-spans within the experiment, which may have been too short for time-based forgetting to occur. We additionally found a more substantial decrease in performance when additional rules are presented, implying that manipulations have a larger negative impact on information retrieval than time. This, in fact, aligns with the concept of *retroactive interference*, as new rules make previous ones vulnerable to decay (Ricker, Nieuwenstein, Bayliss, & Barrouillet, 2018; Dames & Oberauer, 2022).

Since the three basic operators, addition, contraction and revision, appeared to be the most distinctive building blocks, we modeled them using linear regression based on features describing the trial with respect to a rule (i.e., how many shapes are relevant for the rule, if there are any) and with respect to the rule system itself (i.e., how many rules are active at the time). For the models, we found a strong dependency on the trial only, across all operators. The strong dependency of the model to the structure of the trial itself (rather than to the rule system and memory operations) highlights the problem to discern the effects of manipulations of a rule system from the application of the rule itself. Whereas investigations of forgetting in declarative memory can directly rely on recognition of items, investigations of procedural memory often needs application of rules as an intermediate step, thereby assessing forgetting indirectly. While being a drawback for investigating cognitive operations, it can be an advantage for applications, since real-world rule systems are not occurring free of the specific instances and contexts.

Acknowledgments

This project has been funded by a grant to MR in the DFG-projects 529624975, 427257555 and 318378366.

References

Alchourrón, C., Gärdenfors, P., & Makinson, D. (1985). On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, *50*(2), 510–530.

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

Beierle, C., Kern-Isberner, G., Sauerwald, K., Bock, T., & Ragni, M. (2019). Towards a general framework for kinds of forgetting in common-sense belief management. *Künstliche Intelligenz*, *33*, 57-68. doi: https://doi.org/10.1007/s13218-018-0567-3

Brand, D., Dames, H., Puricelli, L., & Ragni, M. (2022). Rule-based categorization: Measuring the cognitive costs of intentional rule updating. In J. Culbertson, A. Perfors, H. Rabagliati, & V. Ramenzoni (Eds.), *Proceedings of the 44th annual meeting of the cognitive science society* (pp. 2810–2817).

Dames, H., Brand, D., & Ragni, M. (2022). Evidence for multiple mechanisms underlying list-method directed forgetting. In J. Culbertson, A. Perfors, H. Rabagliati, & V. Ramenzoni (Eds.), *Proceedings of the 44th annual conference of the cognitive science society.* Retrieved from `https://escholarship.org/uc/item/46921378`

Dames, H., & Oberauer, K. (2022). Directed forgetting in working memory. *Journal of Experimental Psychology*, *151*(12).

Delgrande, J. (2014). Towards a knowledge level analysis of forgetting..

Gollwitzer, P. M., & Brandstätter, V. (1997). Implementation intentions and effective goal pursuit. *Journal of Personality and Social Psychology*, *73*(1), 186–199. doi: 10.1037/0022-3514.73.1.186

Maddox, W. T., Ashby, F. G., & Bohil, C. J. (2003). Delayed feedback effects on rule-based and information-integration category learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *29*(4), 650.

Maddox, W. T., Ashby, F. G., Ing, A. D., & Pickering, A. D. (2004). Disrupting feedback processing interferes with rule-based but not information-integration category learning. *Memory & Cognition*, *32*(4), 582–591.

Oettingen, G., & Gollwitzer, P. M. (2010). Strategies of setting and implementing goals: Mental contrasting and implementation intentions. In J. E. Maddux & J. P. Tangney (Eds.), *Social psychological foundations of clinical psychology* (pp. 114–135). New York: Guilford Press.

Ricker, T., Nieuwenstein, M. R., Bayliss, D. M., & Barrouillet, P. (2018). Working memory consolidation: Insights from studies on attention and working memory: An overview of working memory consolidation. *Annals of the New York Academy of Sciences*, *1424*(1), 8–18. doi: 10.1111/nyas.13633

Hey Pentti, We Did It Again!: Differentiable Vector-symbolic Types that Prove Polynomial Termination

Eilene Tomkins-Flanagan (eilenetomkinsflanaga@cmail.carleton.ca)
Connor Hanley (connorhanley@cmail.carleton.ca)
Mary Alexandria Kelly (mary.kelly4@carleton.ca)
Department of Cognitive Science, Carleton University
1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada

Abstract

We present a typed computer language, Doug, in which all typed programs may be proved to halt in polynomial time, encoded in a vector-symbolic architecture (VSA). Doug is just an encoding of the light linear functional programming language (LLFPL) described by Schimanski (2009, ch. 7). The types of Doug are encoded using a slot-value encoding scheme based on holographic declarative memory (HDM; Kelly, Arora, West, & Reitter, 2020). The terms of Doug are encoded using a variant of the Lisp VSA defined by Tomkins-Flanagan and Kelly (2024). Doug allows for some points on the embedding space of a neural network to be interpreted as types, where the types of nearby points are similar both in structure and content. Types in Doug are therefore learnable by a neural network. Following Chollet (2019), Card, Moran, and Newell (1983), and Newell and Rosenbloom (1981), we view *skill* as the application of a procedure, or program of action, that causes a goal to be satisfied. Skill acquisition may therefore be expressed as *program synthesis*. Using Doug, we hope to describe a form of learning of skilled behaviour that follows a human-like pace of skill acquisition (i.e., *substantially faster than brute force*; Heathcote, Brown, & Mewhort, 2000), exceeding the efficiency of all currently existing approaches (Kaplan et al., 2020; A. L. Jones, 2021; Chollet, 2024). Our approach brings us one step closer to modeling human mental representations, as they must actually exist in the brain, and those representations' acquisition, as they are actually learned.

Keywords: polynomial time type system; representation learning; vector-symbolic architecture

Chollet (2019) proposed a novel model for intelligence: an *intelligent agent* does not just *solve problems* in its environment in service to its goals (as we might infer it does in the Markov decision process model found in *reinforcement learning*, Sutton & Barto, 2018, pp. 46-53). Instead, an intelligent agent *acquires the skills necessary to the task*, and uses its *acquired skills* to solve problems in service to its goals. Chollet's model of "skill" is an extremely general one. In his view, a skill is a *program*: a procedure made up of well-defined steps, which, when followed, will transform the environment from an initial state into a desired state. By separating the intelligent agent into an *intelligent system* that generates skills, and *skill programs* that realize the agent's competencies, Chollet frames the problem of intelligence as one of an ability to acquire skills from sparse examples in a great variety of problem types. Following Chollet, a *greater intelligence* within some scope of problems is one that can acquire the skills to solve problems in that scope with a minimum of foreknowledge and training, and a *more general intelligence* is intelligent, up to some degree, in a greater variety of scopes, more exhaustive of the space of possible problems.

The Chollet model may have a familiar ring for cognitive modelers, however. The *Goals, Operators, Methods, and Selection rules* (GOMS; Card et al., 1983, ch. 5) model describes skilled behaviour as a deployment of procedures that will tend to satisfy an agent's goals. In GOMS, an agent with some *goal G* uses its *selection rules* to choose a *method* that, the agent expects, will cause *G* to be satisfied, if followed. A *method* is a procedure in the sense we used above: a sequence of well-defined steps. In the case of GOMS, the steps are *operators*, discrete actions the agent can take that transform the state of the environment, or the agent's internal state. Following the procedure laid out by the selected method, the environment is transformed into a state whereby the goal *G* is satisfied. Newell (1990) situates GOMS within the *Soar* cognitive architecture, stating that GOMS "posits a set of mechanisms and structures" (p. 285) that can be formalized with the Soar architecture, and goes on to express skill acquisition in Soar (ch. 6.5) as a kind of search for increasingly efficient procedures that satisfy goals.

Chollet refers to the problem of acquiring skills as "program synthesis" (p. 30), that is, search for a procedure, with some given effect. Hutter (2005, ch. 1.5) showed that synthesizing an optimal program that *predicts* or *represents* the environment allows an agent to behave in an optimally goal-directed (i.e., rational) way, with minimal extra machinery. In this way, the problems of *skill acquisition* and *representation learning* can be thought of as transformable into one another.

Hutter, unfortunately, also finds (ch. 1.7), that program synthesis is profoundly nontrivial. In fact, finding a program that optimally predicts an arbitrary environment is incomputable, because it involves comparing programs in light of their computational effects, and some programs cannot be proven either to eventually halt or run forever (Turing, 1936), so their computational effects cannot be known either until they halt or until the end of time, whichever comes first. Even if we restrict ourselves to considering only programs up to some maximum finite length, if we are to be certain we have found the optimal representation, we must do an ugly exhaustive search through all possible programs up to our maximum length (which is intractable, in the sense of taking an exponentially long amount of time as a function of the maximum length) and, for each program, find its shortest representation that halts within some maximum time (which is also intractable). Hutter's finding of a doubly-exponential rate of

skill acquisition is consistent with the "power law" described by Newell and Rosenbloom (1981), however, Heathcote et al. (2000) show that their law is an artifact of population averaging, and humans in fact acquire skill *exponentially faster* than Newell and Rosenbloom believed. Unsurprisingly so, as Newell and Rosenbloom modeled skill acquisition as a simple, heuristic-informed (but in the worst case, like Hutter's model, brute-force) search, and managed to satisfy their power law. It seems obvious that, in order to be useful, a method of program synthesis that exhibits intelligence should be substantially better than this sort of brute-force approach.

Chollet (2019)'s ARC task was designed to demonstrate the efficiency of human intelligence relative to existing AI approaches, and the importance of efficient program synthesis. A model approaching the ARC task that showed some early success was DreamCoder (Ellis et al., 2021), which used a similar method to Hutter's, albeit synthesizing programs using a form of heuristic informed search involving a neural network. DreamCoder has since been surpassed by the GPT o-series models (Chollet, 2024), but Chollet (2025) notes that both approaches seem to behave like exhaustive search:

> It's always possible to ... logarithmically improve your performance by throwing more compute at the problem. And of course this is true for o1, but even before that it was also true for brute force program search systems. Assuming you have the right [domain-specific language], then extremely crude, basic, brute-force program iteration can solve ARC at human level. (49:55)

We would like to constrain program synthesis so that search is restricted only to programs likely to be useful to solving a given problem. If we make good assumptions about the set of programs that are to be searched over, then, for many problems, searching through the constrained set of possible programs for their solution should become tractable. As a first pass, we should prohibit from consideration intractable programs, as we do not want to bother trying to evaluate them to discover their computational effects. This prohibition is achieved using a polynomial time type system (Girard, 1998).

A *type system* is a programming language that assigns, to each expression in a program, a *type* that describes what kind of data it is, if it is a variable, and what it operates on and produces, if it is a function, accompanied with a set of *inference rules* for how to demonstrate that some expression has some type. A *polynomial time type system* is a type system that prohibits programs which do not have polynomial time complexity. A polynomial time type system is therefore not Turing-complete, but the set of problems polynomial time type systems are capable of representing still includes essentially all useful programs. The type system we will be considering is the *Light Linear Functional Programming Language* (LLFPL; Schimanski, 2009, ch. 7).

How can we make use of a polynomial time type system to constrain program synthesis? LLFPL, as an example, encodes the maximum recursion depth of a function in its type. We envision that types should be learnable, such that a learning agent should be able to acquire the ability to guess at the sort of structure a skill program should have, before it sets to work determining the program's specifics. The agent may need to revise its guess, but it should at least be capable of acquiring the ability to make good guesses. When a learning agent makes good guesses as to the structure of the program it should be synthesizing, its search is immediately restricted to just the programs of the appropriate structure. This set may still be quite large, but, if the appropriate structure restricts recursion depth, it is definitely much smaller than the set of all programs, or even all well-typed programs in the type system.

Our goal, then, is to make LLFPL *learnable by a neural network*. To do so, we will need to make it possible to express types as *points in a vector space*, and we should make it so that structurally similar types are nearby in the space, such that a small change in position in the space connotes a small change in the structure of the type at the new position. In a word, we would like our types to be *differentiable*.

Any arbitrary syntactic structure (of which types are a sort) can be encoded over a vector space in this way (Tomkins-Flanagan & Kelly, 2024), using a *vector-symbolic architecture* (VSA). In a VSA, we expect structures composed of distinct elements to be nearby in the vector space if (1) they have the same structure, and (2) the elements of which they are composed are also nearby. We will go beyond Tomkins-Flanagan and Kelly, however, and take proper advantage of the features of a VSA to make distinct but similar structures spatially nearer to one another. We define a language, based on Tomkins-Flanagan and Kelly's Lisp VSA and LLFPL, called the *vector-symbolic Lisp representation of the light linear functional programming language* (VSLRLLFPL), or Doug[1], as the other name is very long.

A body of many organs

Doug builds on the work of Tomkins-Flanagan and Kelly (2024) in order to (1) allow points on a neural network's *embedding space* to encode *systematically* decodable types, (2) constrain possible programs typed by any point on the embedding space to include only those that halt in polynomial time, and (3) for nearby points to encode types that are both *structurally similar* and *comprised of similar elements*. Therefore, the surface induced by the embedding space and some loss function on decoded types will be differentiable, where traversing the surface by *gradient descent* will cause relatively smooth changes in the structure and content of types encoded at nearby points, even if not all points are decodable. In other words, *the structure and content of types in Doug is learnable by a neural network*.

In order to achieve the three preceding goals, we must first select a polynomial time type system; as above, we use LLFPL. We will then recapitulate Tomkins-Flanagan and Kelly's definition of a VSA for the benefit of the novice

[1]Our implementation may be found at `https://github.com/eilene-ftf/doug`

reader, supplementing the discussion with the additional elements necessary to encode Doug, including Kymn et al. (2023)'s *residue numbers*. We will then select a VSA-based encoding scheme whereby *syntactically* similar structures are encoded as *spatially* similar vectors, if they are similar in content. The encoding scheme we choose is the one employed by Kelly et al. (2020)'s *holographic declarative memory* (HDM). The preceding parts will come together so that, in the next section, we can encode the *types* of Doug using the slot-value encoding of HDM with residue numbers, and the *terms* using a variant of Tomkins-Flanagan and Kelly (2024)'s Lisp VSA.

Lightly linear flipout

In this section, we will introduce some of the history and motivations behind linear type systems, polyonomial time type systems, as well our language of choice, the *Light Linear Functional Programming Language* (Schimanski, 2009).

History and intuitions A type system is a full language specification that constrains the kinds of expressions in the language based on the notion of a *type*. Types introduce primitives into the system which regiment the kinds of things that our language deals with and puts constraints on what we can do with those things (Nederpelt & Geuvers, 2014).

One constraint that we might like to put on our programming language is that to constrain the amount of usages of some item. Consider the following: suppose we are writing a program for a resource-constrained old computer, and we want to ensure both (a) efficient (i.e., as little as possible) usage of the limited amount of memory available, and (b) ensure that all memory that we allocate for the program, if we do so, is neatly "put back" in place as quickly as possible. Motivations like these inspired Girard (1987) to formulate *Linear Logic* (LL) which captures the aforementioned goals. Simply put, it does this by restricting the number of usages of items by requiring that they are used *exactly* once in programs. But we say that a type system is *affine* if and only if it requires that terms be used *at most* once.

LLFPL Schimanski (2009) is a systematic study of polynomial time type systems. The language that we will be encoding here is Schimanski (2009)'s own contribution: *Light Linear Functional Programming Language* (LLFPL$_!$), which extends the *Linear Functional Programming Language* (LFPL; Hofmann, 2003) by combining it with elements from *Light Linear Logic* (LLL; Girard, 1998). Here we will lay out the language definition in the standard way, using Backus-Naur form. After, we will give a natural language explanation of what the constants are meant to denote.

Definition 1 (Types of LLFPL; Levels of types) *The set of* types *of LLFPL are defined by the following expression,*

$$\sigma,\tau ::= B^n \mid \sigma \multimap \tau \mid \sigma \otimes \tau \mid L^n(\sigma) \mid !^n\sigma \mid \diamond^n. \tag{1}$$

The level *of a type is defined recursively by the function,*

$$\ell(\rho) := \begin{cases} n \text{ if } \rho \in \{B^n, L^n(\sigma), !^n\sigma, \diamond^n\}, \\ \min\{\ell(\sigma), \ell(\tau)\}, \text{ otherwise.} \end{cases} \tag{2}$$

More naturally, B^n is the type of *Booleans*, $\sigma \multimap \tau$ the *linear function* type. $\sigma \otimes \tau$ is the tuple type of pairs, $L^n(\sigma)$ is the type of lists of type σ. $!^n\sigma$ is a modal operator denoting that the embedded type σ can be used "arbitrarily often" (Schimanski, 2009). $\diamond^n$ is a *credit* type from Hofmann (2003), which is used to limit recursion depth.

Definition 2 (The constants of LLFPL) *The* constant *terms of LLFPL, which are* constructors *and* destructors *for the types given in Definition 1.*

$$\mathbf{tt}^n, \mathbf{ff}^n : B^n, \tag{3}$$
$$\mathbf{Case}^n_\sigma : B^n \multimap \sigma \multimap \sigma \multimap \sigma, \tag{4}$$
$$\mathbf{Case}^n_{\tau,\sigma} : L^n(\tau) \multimap (\diamond^n \multimap \tau \multimap L^n(\tau) \multimap \sigma) \multimap \sigma \multimap \sigma, \tag{5}$$
$$\mathbf{cons}^n_\tau : \diamond^n \multimap \tau \multimap L^n(\tau), \tag{6}$$
$$\mathbf{nil}^n_\tau : L^n(\tau), \tag{7}$$
$$d^n : \diamond^n, \tag{8}$$
$$\otimes^n_{\tau,\rho} : \tau \multimap \rho \multimap \tau \otimes \rho, \tag{9}$$
$$\pi^n_\sigma : \tau \otimes \rho \multimap (\tau \multimap \rho \multimap \sigma) \multimap \sigma. \tag{10}$$

Where for Eq.'ns (4-5), we have the constraint that $\ell(\sigma) \geq n$, for Eq. (9) that $\ell(\sigma \otimes \tau) = n$, and finally for Eq. (10) that $\ell(\sigma) \geq \ell(\tau \otimes \rho) = n$.

Here, the intuitive "meaning" behind each constant is listed: $\mathbf{tt}^n, \mathbf{ff}^n$ are *true* and *false* constants, or $\top$ and $\bot$. $\mathbf{Case}^n_\sigma$ and $\mathbf{Case}^n_{\sigma,\tau}$ are *destructors* for boolean types and list types (respectively); $\mathbf{cons}^n_\tau$ and $\mathbf{nil}^n_\tau$ are the constructor and base element of list types; d^n is a *chit* of the *credit type*, a sort of token you have to give to recursive procedures that is consumed on usage in order to limit the depth of recursion; $\otimes^n_{\tau,\rho}$ is the constructor for tuple types; and finally, π^n_σ is the *projection function*, which is the destructor for tuples types.

The language is not just a collection of types and constants, but also terms which form the object-level of the language. Terms are what are used to express the procedures which an interpreter, computer, or agent, follows.

Definition 3 (Terms of LLFPL) *The* terms *of LLFPL are defined inductively,*

$$s,t ::= x : \tau \mid c \mid (\lambda x : \tau.t) \mid (t\ s) \mid !^n\boxed{x = \{s\}_{!^n} \text{ in } t} \mid !^n\boxed{t} \mid \{t\}_{!^n} \mid \left(s \overset{n^{x_1}}{\triangleleft_{x_2}} t\right). \tag{11}$$

where we have types τ, constants c, natural numbers n, and variables $x_1, \ldots, x_m$. Constants of LLFPL are terms.

More naturally, x's are variables, c's constants, $(\lambda x : \tau.t)$ is a λ-abstraction (Nederpelt & Geuvers, 2014, p. 2), and $(t\ s)$ an application of a function t to s. The special terms here are the *boxed terms* $!^n\boxed{\cdot}$. A boxed term is "closed" (i.e., has no free variables), except for those bound by terms in holes $\{\cdot\}_{!^n}$ (Schimanski, 2009, p. 200). A term t enclosed by a box $!^n\boxed{t}$ has level $n+1$. When a hole is filled in, as in $!^n\boxed{x = \{s\}_{!^n} \text{ in } t}$, we bind x to the value s in the term t, similarly to a function application. When a boxed term is en-

closed in a hole of the same level $\{!^n\boxed{t}\}_{!n}$, the bang and box are eliminated, and t is lowered from level $n+1$ to level n.

A type system is not complete without accompanying inference rules, which are a collection of rewriting rules specifying when and under what conditions terms can be properly said to have some type. We will not be enumerating the inference rules of the type system. For a complete presentation of LLFPL's inference rules see Schimanski (2009, pp. 209-210)

How LLFPL is polynomial Schimanski (2009, ch. 7.3.5) proves that LLFPL can only express programs that halt in polynomial time. Many useful functions, like *quicksort*, may be expressed in LLFPL, but intractable functions cannot be. LLFPL achieves its polynomial restriction by a careful interplay of the credit and boxed expressions during the evaluation of recursive expressions. The key evaluation rule to consider is that defined for *folded* expressions, where a function mapped over a list has a variable bound to a holed term.

$$\Big((\mathbf{cons}^{\tau}_{n+1} t^{\diamond^{n+1}}\, v\ l)\ f[z := \{r\}_{!n}]\ g\Big) \mapsto^{l}_{n+1} \Big(r \overset{n\, r_1}{\triangleleft_{r_2}} (f[z := \{r_1\}_{!n}]\ t\ v\ (l\ f[z := \{r_2\}_{!n}]\ g))\Big) \tag{12}$$

The left side of the above should be read as the *application of a list* consisting of a head v and a tail l to a recursive case f and a base case g, where the variable z is bound to the holed value r in f. When a list is "applied", it just means that f is to be folded over each value of the list in turn, until the base case. On the right hand side, we have that the *multiplexer* $\triangleleft$ copies r into r_1, r_2, which must be done explicitly since copying is restricted. Then, f is applied to t, the credit, v, the head, and the result of the recursive map over the tail l.

Because applying f consumes a chit, which are stored in lists, there must be linearly many calls to f in the length of the list. Iteration calls f multiple times, but it can't make exponentially proliferating recursive calls. A variable in f may be bound to another recursive term, allowing nested recursion, but since it's a holed term, it must be one level below f. As a result, linear recursive calls can be nested only up to the maximum level of a term. *In order to increase the maximum polynomial order of a term, one must increase its level.*

What's a VSA again?

Tomkins-Flanagan and Kelly (2024) define a VSA as:

> A vector-symbolic architecture is an algebra (i.e., a vector space with a bilinear product),
>
> 1. that is closed under the product $\otimes : V \times V \to V$ (i.e., if $u \otimes v = w$, then $u, v, w \in V$)
> 2. whose product has an "approximate inverse" $\overline{\otimes}$ that, given a product w and one of its operands u or v, yields a vector correlated with the other operand
> 3. for which there is a dogma for selecting vectors from the space to be treated as atomic "symbols" (yielding themselves, thereby, to syntactic manipulations defined in terms of the algebra),
> 4. that is paired with a memory system $\mathcal{M}$ that stores an inventory of known symbols for retrieval after lossy operations (e.g., involution), that can be recalled from $\mathcal{M}(p)$, and which is appendable $\mathcal{M} \twoheadleftarrow t$, and
> 5. possesses a measure of the correlation (a.k.a., similarity) of two vectors, $\mathbf{sim}(u, v) \in [-1, 1]$, where 1 and -1 imply that u, v are colinear, 0 that they are linearly independent.

Certain VSAs relax the above properties, but all behave in a manner that approximates these properties. Tomkins-Flanagan and Kelly also show that VSAs are *Cartesian closed* under these properties, meaning that a VSA can express an arbitrary *Turing-complete language* over vectors of fixed dimension (so long as the memory may be arbitrarily large). For our convenience, we extend the definition of a generic VSA with *permutations*, used in HDM, as well as a *second* product operator $\star$ and *resonator networks*, used in the *residue numbers* we employ to encode the natural numbers.

A *permutation* $\mathbf{P}_c(v)$ is a function that reorders the dimensions of a vector v. That is, for a finite vector $v \in V$, $\mathbf{P}_c(v) = v' \in V$ where $v'_j = v_i$, where all values of v_i are mapped to exactly one v'_j. Permutations are invertible, so $\mathbf{P}_c^{-1}(\mathbf{P}_c(v)) = v$. The *second product* $\star$ behaves as $\otimes$, except that $a \star b \neq a \otimes b$ in general, and each product has a different multiplicative unit. Given some composite representation $v = \bigotimes_{i=1}^{k} a_i$, where $a_i \in A_i$, a resonator network decomposes v into a tuple of the elements of which it is composed: $R(v, A_1, ...A_k) = (a_1, ..., a_k)$.

Permutative concerns Permutations are typically applied together with the first product operator $\otimes$ in order to achieve *asymmetric binding*. That is, where $a \circledast b = \mathbf{P}_{\text{right}}(a) \otimes \mathbf{P}_{\text{left}}(b)$ for constant permutations $\mathbf{P}_{\text{left}}, \mathbf{P}_{\text{right}}, \mathbf{P}_{\text{left}} \neq \mathbf{P}_{\text{right}}$, we have that $a \circledast b \neq b \circledast a$. This allows for the inductive encoding of sequences $a \circledast (b \circledast (c \circledast ...))$.

Residual notes Residue numbers use complex-domain holographic reduced representations (Plate, 2003). We will try to make the treatment as generic as possible for the purpose of our formalism. Given a function ζ that generates a VSA representation of a natural number n, $\zeta(n) = v$, we define the sum as $\zeta(n+m) = \zeta(n) \otimes \zeta(m)$ and the product as $\zeta(nm) = \zeta(n) \star \zeta(m)$. Numbers are encoded using a sum of modular vector representations, so $\zeta(n) = z_p(n) \otimes z_q(n) \otimes z_r(n) \otimes ...$, where $p, q, r, ...$ are positive coprime integers, and $z_s(n) = z_s(n \mod s)$. See Kymn et al. (2023).

Resonant decoding As above, a resonator network is defined as $R(\zeta(n), P, Q, R, ...) = (z_p(n), z_q(n), z_r(n), ...)$ for moduli $p, q, r, ...$ Each set S used in the decoding contains all the possible values of the corresponding function z_s, so $S = \{z_s(1), ..., z_s(s)\}$, since $z_s(n) = z_s(n \mod s)$. When a numeric representation is decoded into its modular constituents, the exact value n can be decoded. Given we decode the tuple $(z_p(n \mod p), z_q(n \mod q), z_r(n \mod r), ...)$, we may further infer that the tuple of natural numbers ($n \mod p$, n

mod q, $n \mod r, \ldots$) identifies the encoded number n, That tuple uniquely encodes n up to the least common multiple of $p, q, r, \ldots$ See Frady, Kent, Olshausen, and Sommer (2020).

Certain holographic declarations

Kelly et al. (2020) describes Holographic Declarative Memory (HDM), a *declarative memory* module for the ACT-R cognitive architecture (Anderson, 1993) that uses a VSA (specifically, holographic reduced representations; Plate, 2003) to encode *memory chunks*. In ACT-R, a chunk is a data structure that can be held in memory. In ACT-R, declarative memory contains both semantic and episodic memory, and, when *probed* with the appropriate cue, will *recall* whatever *chunk* it stores that is most similar to the cue.

There are two types of information stored in an ACT-R chunk. They are *sequential* information, and *slot-value* information. Each kind of information consists of *symbols* in the sense intended by Newell (1980); namely, they *refer to* some *meaning* (i.e., another object of cognition), and that having a symbol confers *distal access* to whatever meaning it symbolizes. *Sequential* information is coded in a list-like format, where the symbols are ordered, and the position of each symbol matters. In the *slot-value* format, symbols are stored in named, unordered *slots*, and the chunk can be decomposed by retrieving symbols from it in some known slot. We are only interested in the slot-value encoding.

In HDM, each slot of a chunk has a permutation associated with it, $\mathbf{P}_{\text{slot}}$. Objects of the same kind are stored in chunks that contain the same slots, so, if one were representing shapes of various colours and sizes, one might have chunks like `(shape:circle colour:red size:large)` or `(shape:square colour:blue size:small)`. A special placeholder value Φ is stored and held constant across all chunks. HDM is interested in the semantic content of values, for use in retrieval from declarative memory. As declarative memory is cued with chunks similar to what was stored, and probing it yields a chunk similar to the probe, *values* are thought of as answers to the question "what goes in the empty spot of my incomplete chunk?" Given one is storing information about redness, instead of storing information about the value *red* directly, HDM will store information about the context in which *red* appears, and use a placeholder to stand in for *red*. That is, supposing one has a large red circle in working memory, and one is storing information about redness, one stores $\mathbf{q}_{\text{red}} = \mathbf{P}_{\text{colour}}(\Phi) \otimes \mathbf{P}_{\text{shape}}(v_{\text{circle}}) \otimes \mathbf{P}_{\text{size}}(v_{\text{large}})$. The resulting stored value corresponding to the colour red is just the sum of all the stored contexts in which red occurs.

Because we can think of $\mathbf{q}_{\text{red}}$ as a question to which the colour red is an answer (i.e., an incomplete chunk where red fills the colour slot), a chunk may be constructed as $\mathbf{c}(\text{large red square}) = \mathbf{P}_{\text{colour}}(v_{\text{red}}) \otimes \mathbf{P}_{\text{shape}}(v_{\text{circle}}) \otimes \mathbf{P}_{\text{size}}(v_{\text{large}})$, and, should one be interested in describing a neural network that is able to complete type signatures given partial information, one can train a network to associate v_{red} with a distribution of questions like $\mathbf{q}_{\text{red}}$. For our purposes, we are more interested in chunks $\mathbf{c}$. In the simple scheme presented by Kelly et al. for HDM, chunks that are alike in structure and content will be spatially nearby. However, the HDM scheme is slightly too constraining, as chunks that are alike in structure, and similar in content, but unalike in one value, will be very dissimilar. Accordingly, we iterate on the HDM chunk representation by following a BEAGLE-like formula (M. N. Jones & Mewhort, 2007), and representing a chunk as the sum of the products of all subsets the chunk. In the preceding example, with $\mathcal{C} = \mathcal{P}(\{\text{size} : \text{large}, \text{shape} : \text{circle}, \text{colour} : \text{red}\})$, $\mathbf{c}(\text{size} : \text{large} \quad \text{colour} : \text{red} \quad \text{shape} : \text{square}) = \sum_{c \in \mathcal{C}} \bigotimes_{\text{slot:value} \in c} \mathbf{P}_{\text{slot}}(v_{value})$, where $\mathcal{P}$ denotes a powerset. Chunks are normalized to a magnitude of 1. Because the unary subsets of a chunk are encoded in its representation, a value may be decoded from a chunk $\mathbf{c}$ by $\mathcal{M}(\mathbf{P}_{slot}^{-1}(\mathbf{c}))$, provided values are stored in the memory system.

VSLRLLFPL, or, Doug

Tomkins-Flanagan and Kelly (2024) provide a method for encoding any arbitrary syntax into a VSA using traditional *role-filler pairs* commonly used in both symbolic (Ritter, Tehranchi, & Oury, 2019) and vector-symbolic systems (Plate, 2003; Smolensky, 1990). Therefore, in order to capture the polynomial time type system within the VSA, we propose the following encoding.

Definition 4 (Doug Types) *For the following definition symbols marked in* ***bold*** *will denote vector-symbols of a sufficiently high dimensionality D sampled according to the dogma of the chosen VSA, except* $\mathbf{c}$*, which denotes the chunk constructor. The* tags *of the encoding will be* **Boolean**, **Map**, **Tuple**, **List**, **Bang**, *and* **Credit**.

Recalling the encoding scheme we derived from HDM above, a chunk of slot-value pairs is encoded as, $\mathbf{c}(\text{slot}_1 : \mathbf{value_1} \quad \text{slot}_2 : \mathbf{value_2} \ldots) = \sum_{c \in \mathcal{C}} \bigotimes_{\text{slot:value} \in c} \mathbf{P}_{\text{slot}}(\mathbf{value})$*, where* $\mathcal{C} = \mathcal{P}(\{\text{slot}_1 : \text{value}_1, \text{slot}_2 : \text{value}_2, \ldots)$*, we construct the following types inductively.*

For each type in Def. 1, we proceed step-wise with an encoding function:

$$
\begin{aligned}
\mathit{boolean}(n) &:= \mathbf{c}(\text{kind} : \mathbf{Boolean} \quad \text{type} : \mathbf{B} \quad \text{level} : n),\\
\mathit{map}(d,c) &:= \mathbf{c}(\text{kind} : \mathbf{Map}\\
&\qquad \text{type} : \mathbf{c}(\text{dom} : d \quad \text{codom} : c), \quad \text{level} : n),\\
\mathit{tuple}(l,r) &:= \mathbf{c}(\text{kind} : \mathbf{Tuple}\\
&\qquad \text{type} : \mathbf{c}(\text{left} : l \quad \text{right} : r), \quad \text{level} : n),\\
\mathit{list}(n,s) &:= \mathbf{c}(\text{kind} : \mathbf{List} \quad \text{type} : s, \quad \text{level} : n),\\
\mathit{bang}(n,s) &:= \mathbf{c}(\text{kind} : \mathbf{Bang} \quad \text{type} : s, \quad \text{level} : n),\\
\mathit{credit}(n) &:= \mathbf{c}(\text{kind} : \mathbf{Credit} \quad \text{type} : \mathbf{D}, \quad \text{level} : n).
\end{aligned}
$$

Where n uniformly denotes a residue natural number.

The encoding allows for (1) structuring the different sub-elements of the types in a compositional manner, (2) querying a representation for a specific sub-element, and (3) a notion of *similarity* between two types.

We encode the constants found in Def. 2 as follows:

Definition 5 (Doug Constants) *We maintain the same convention as Def. 4 for denoting vector-symbols. Let us have tag vector-symbols* **TT**, **FF**, $\mathbf{Case_{bool}}$, $\mathbf{Case_{list}}$, **Cons**, **Nil**, **Dollar**, **Pair**, *and* **Proj**.

We proceed step-wise for each item in Def. 2,

$$
\begin{aligned}
tt(n) &:= \mathbf{TT} + (\mathbf{level} \otimes n),\\
ff(n) &:= \mathbf{FF} + (\mathbf{level} \otimes n),\\
case_{bool}(n,s) &:= \mathbf{Case_{bool}} + (\mathbf{level} \otimes n) + (\mathbf{type} \otimes s),\\
case_{list}(n,t,s) &:= \mathbf{Case_{list}} + (\mathbf{level} \otimes n) + (\mathbf{from} \otimes t) + (\mathbf{to} \otimes s),\\
cons(n,t) &:= \mathbf{Cons} + (\mathbf{level} \otimes n) + (\mathbf{type} \otimes t),\\
nil(n,t) &:= \mathbf{Nil} + (\mathbf{level} \otimes n) + (\mathbf{type} \otimes t),\\
dollar(n) &:= \mathbf{Dollar} + (\mathbf{level} \otimes n),\\
pair(n,l,r) &:= \mathbf{Pair} + (\mathbf{level} \otimes n) + (\mathbf{left} \otimes l) + (\mathbf{right} \otimes r),\\
proj(n,s) &:= \mathbf{Proj} + (\mathbf{level} \otimes n) + (\mathbf{type} \otimes s),
\end{aligned}
$$

where again n is some natural number encoding.

Constant terms of the language represent *constructors* and *destructors* of types, which are ways we can express type introduction and elimination (Univalent Foundations Program, 2013, pg. 27).

Types and constants do not make up the whole language: we must also have a way of encoding arbitrary expressions.

Definition 6 (Doug Terms) *We adopt the same vector-symbolic conventions above and sample vectors of the same dimensionality. Following Def. 3, let the tag symbols be* **annotation**, **Const**, **Lambda**, **App**, **Box**, **Brackets**, *and* **Sub**.

Step-wise, the encoding of terms is as follows:

$$
\begin{aligned}
annotation(x,\tau) &:= \mathbf{Annotation} + (\mathbf{var} \otimes x) + (\mathbf{type} \otimes \tau),\\
const(c) &:= \mathbf{Const} + (\mathbf{val} \otimes c),\\
\\
lambda(x,\tau,t) &:= \mathbf{Lambda} + (\mathbf{var} \otimes x) + (\mathbf{type} \otimes \tau)\\
&\quad + (\mathbf{body} \otimes t),\\
app(t,s) &:= \mathbf{App} + (\mathbf{rator} \otimes t) + (\mathbf{rand} \otimes s),\\
box(n,x,s,t) &:= \mathbf{Box} + (\mathbf{let} \otimes x) + (\mathbf{this} \otimes s)\\
&\quad + (\mathbf{that} \otimes t) + (\mathbf{level} \otimes n),\\
brace(n,t) &:= \mathbf{Brace} + (\mathbf{level} \otimes n) + (\mathbf{term} \otimes t),\\
sub(s,n,x_1,x_2,t) &:= \mathbf{Sub} + (\mathbf{this} \otimes t) + (\mathbf{that} \otimes s)\\
&\quad + (\mathbf{level} \otimes n) + (\mathbf{x1} \otimes x_1) + (\mathbf{x2} \otimes x_2),
\end{aligned}
$$

where n is some natural number encoded in HRR.

Discussion

Doug allows us to encode types over a vector space. We view types as a sort of *constraint* on program synthesis; given a type, one narrows a search space of possible programs. If a type is chosen reasonably, and the type system is sufficiently constraining, the search space may be constrained to such an extent that searching for a program satisfying some goal can be done in polynomial, not exponential time. Finding an optimal program, we expect, will remain computationally hard. But finding any program that satisfies a goal, and furthermore, constraining a search to only consider programs of constant behaviour should not be, as, during skill acquisition, humans tend to do so with ease: human skill acquisition is not doubly exponential, connoting exhaustive search both for more efficient programs and programs of the correct behaviour (as suggested by Hutter, 2000, ch. 1), but singly-exponential, connoting a hard search for more efficient procedures, but an easier search for correct behaviour (Heathcote et al., 2000).

Dehaene, Al Roumi, Lakretz, Planton, and Sablé-Meyer (2022) found that the neural representations for at least some simple skills seem to have the information content of optimal programs. There are two suggestions that follow from their finding: First, that human mental representations must be expressive enough to store arbitrary programs, at least up to some maximum complexity permitted by memory capacity. Second, humans are fairly good at finding optimal representations, up to the limits of what is tractable. This result is unsurprising: Hutter found that, given an optimal representation of the world, rational goal-directed behaviour is simple to achieve. Humans tend to behave rationally, given the resources to do so. Under the commitments of the common model of cognition (Laird, Lebiere, & Rosenbloom, 2017), humans are "boundedly rational", or, one might say, *rational, up to the limits of what is tractable*, and that *the degree of rational behaviour one can achieve is a skill issue*.

Tomkins-Flanagan and Kelly (2024) argued that VSAs provide the machinery necessary to interpret brain states as syntactically structured representations; in other words, if humans solve problems by generating *skill programs* that *may be optimal* for some tasks, human brain states must encode programs. Furthermore, humans must learn to generate those programs efficiently, and, in order to do so, their search for those programs must be constrained to a subset of programs that may be useful. Those constraints must be sufficiently strict to radically accelerate program synthesis relative to brute-force search, and must *also* be interpretable as brain states, and must be learnable, as humans somehow acquire knowledge of the constraints appropriate to novel problems.

Doug is the first step in describing *learnable, provably strong constraints* that might limit the complexity of program synthesis. By constraining a type system to express only programs that run in polynomial time, programs typed with Doug may express only tractable solutions to given problems. However, it remains to be shown which, if any, type systems can make program synthesis polynomial, and whether learning over those types does not take so long as to cancel any benefits gleaned from constraint. Nevertheless, Doug captures important intuitions about what human skill acquisition should be like, and provides a methodology by which future type systems that really constrain program synthesis like humans can be devised.

References

Anderson, J. R. (1993). *Rules of the mind*. Lawrence Erlbaum

Associates, Inc.

Card, S. K., Moran, S. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Lawrence Erlbaum Associates, inc., Publishers.

Chollet, F. (2019). *On the measure of intelligence.* Retrieved from https://arxiv.org/abs/1911.01547

Chollet, F. (2024). *OpenAI o3 breakthrough high score on ARC-AGI-Pub.* https://arcprize.org/blog/oai-o3-pub-breakthrough. (Accessed: 2025-01-22)

Chollet, F. (2025). *François chollet on openai o-models and arc.* Retrieved from https://www.youtube.com/watch?v=w9WE1aOPjHc

Dehaene, S., Al Roumi, F., Lakretz, Y., Planton, S., & Sablé-Meyer, M. (2022). Symbols and mental programs: a hypothesis about human singularity. *Trends in Cognitive Sciences*, *26*(9), 751-766. doi: 10.1016/j.tics.2022.06.010

Ellis, K., Wong, C., Nye, M., Sablé-Meyer, M., Morales, L., Hewitt, L., ... Tenenbaum, J. B. (2021). Dreamcoder: bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation* (p. 835–850). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3453483.3454080

Frady, E. P., Kent, S. J., Olshausen, B. A., & Sommer, F. T. (2020, December). Resonator Networks, 1: An Efficient Solution for Factoring High-Dimensional, Distributed Representations of Data Structures. *Neural Computation*, *32*(12), 2311–2331. doi: 10.1162/neco_a_01331

Girard, J.-Y. (1987). Linear logic. *Theoretical Computer Science*, *50*(1), 1–101.

Girard, J.-Y. (1998). Light Linear Logic. *Information and Computation*, *143*(2), 175–204. doi: 10.1006/inco.1998.2700

Heathcote, A., Brown, S., & Mewhort, D. J. K. (2000, Jun 01). The power law repealed: The case for an exponential law of practice. *Psychonomic Bulletin & Review*, *7*(2), 185-207. doi: 10.3758/BF03212979

Hofmann, M. (2003, May). Linear types and non-size-increasing polynomial time computation. *Information and Computation*, *183*(1), 57–85. doi: 10.1016/S0890-5401(03)00009-9

Hutter, M. (2000). *Towards a universal theory of artificial intelligence based on algorithmic probability and sequential decision theory.*

Hutter, M. (2005). *Universal artificial intelligence*. Springer.

Jones, A. L. (2021). *Scaling scaling laws with board games.* Retrieved from https://arxiv.org/abs/2104.03113

Jones, M. N., & Mewhort, D. J. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological review*, *114*(1), 1-37. doi: 0.1037/0033-295X.114.1.1

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... Amodei, D. (2020). *Scaling laws for neural language models.* Retrieved from https://arxiv.org/abs/2001.08361

Kelly, M. A., Arora, N., West, R. L., & Reitter, D. (2020). Holographic declarative memory: Distributional semantics as the architecture of memory. *Cognitive Science*, *44*(11), e12904. doi: 10.1111/cogs.12904

Kymn, C. J., Kleyko, D., Frady, E. P., Bybee, C., Kanerva, P., Sommer, F. T., & Olshausen, B. A. (2023, November). *Computing with Residue Numbers in High-Dimensional Representation.* arXiv. (arXiv:2311.04872 [cs]) doi: 10.48550/arXiv.2311.04872

Laird, J. E., Lebiere, C., & Rosenbloom, P. S. (2017, 12). 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. doi: 10.1609/aimag.v38i4.2744

Nederpelt, R., & Geuvers, H. (2014). *Type Theory and Formal Proof: An Introduction* (1st ed.). Cambridge University Press. doi: 10.1017/CBO9781139567725

Newell, A. (1980). Physical symbol systems. *Cognitive science*, *4*(2), 135–183.

Newell, A. (1990). *Unified theories of cognition*. Harvard University Press.

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

Plate, T. A. (2003). *Holographic reduced representation*. University of Chicago Press.

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2019). Act-r: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, *10*(3), e1488.

Schimanski, S. (2009). *Polynomial time calculi*. Unpublished doctoral dissertation, Ludwig Maximilian University of Munich.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, *46*(1), 159-216. doi: 10.1016/0004-3702(90)90007-M

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: an introduction, second edition*. Cambridge, MA: MIT Press.

Tomkins-Flanagan, E., & Kelly, M. A. (2024). Hey Pentti, We Did It!: A Fully Vector-Symbolic Lisp. *Proceedings of the 22nd International Conference on Cognitive Modeling*, 65-72. Retrieved from https://github.com/eilene-ftf/holis

Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, *58*(345-363), 5.

Univalent Foundations Program, T. (2013). *Homotopy type theory: Univalent foundations of mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book.

Predicting Human Behavior in a Robot Interruption Task Using a Cognitive Model

Timo Wiesner[1], Rebecca von Engelhardt[1], Olger Siebinga[2], Chenxu Hao[2], Arkady Zgonnikov[2], Nele Russwinkel[1]
rebecca.engelhardt@uni-luebeck.de
[1]University of L beck, Germany
[2]Delft University of Technology, The Netherlands

Abstract

Interruptions are a fundamental challenge requiring individuals to efficiently manage task switching and cognitive load. Artificial Intelligence (AI) assistants aim to mitigate these challenges, yet their effectiveness is often hindered by poorly timed or irrelevant interventions. This study explores the use of cognitive modeling to predict human behavior in interruption-prone environments, with the goal of enhancing AI-driven assistance. A controlled cooking task experiment was conducted, in which participants completed a four-step recipe while interacting with a robotic assistant providing task suggestions with varying degrees of usefulness. An ACT-R-based cognitive model, grounded in the memory for goals framework, was developed to simulate human task-switching behavior. Model validation followed an iterative refinement process, comparing predictions against human data. The findings indicate that the cognitive model closely approximates human task execution times in two conditions with different levels of usefulness of the robot suggestions. The study demonstrates the feasibility of using cognitive models to model and predict human behavior in interruption tasks which could be used to improve human-robot interactions.

Keywords: ACT-R, cognitive models, human-centered technologies, disruptive interrupts, human aware AI

Introduction

Interruptions are an inherent aspect of modern life, requiring individuals to frequently shift their attention between competing tasks. A common example is cooking, where one must manage multiple processes simultaneously, such as stirring a sauce while ensuring that the pasta does not boil over. Such multitasking scenarios require efficient cognitive strategies to maintain progress and minimize errors (Koundal et al., 2024). While some individuals navigate these challenges with ease, others find them cognitively demanding, particularly when confronted with unexpected interruptions. The extent to which an interruption disrupts task performance depends on several factors such as interruption duration (Borst, Taatgen, & Rijn, 2010), task complexity (Speier, Vessey, & Valacich, 2003) or interruption timing (Adamczyk & Bailey, 2004). Given the increasing complexity of daily tasks, understanding and mitigating the disruptive effects of interruptions is crucial for designing intelligent systems that effectively support human users.

A promising approach to alleviating the cognitive demands of multitasking is the integration of Artificial Intelligence (AI) assistants. These systems have the potential to reduce users' mental workload by providing timely suggestions, reminders, or even autonomously handling specific subtasks. However, despite their intended purpose, AI assistants are often perceived as intrusive rather than helpful, sometimes exacerbating cognitive strain instead of alleviating it (Bitkina, Kim, Park, Park, & Kim, 2021). This issue arises particularly when AI systems fail to recognize contextual nuances, leading to poorly timed or irrelevant suggestions. As a result, understanding the interaction between humans and AI better in order to improve it is essential to ensure that these technologies truly enhance user experience rather than hinder it.

A solution to this problem could be the use of cognitive models, which can simulate and predict human behavior and could therefore be used by robots to predict when and how suggestions are perceived as helpful or disruptive. Cognitive architectures such as Adaptive Control of Thought-Rational (ACT-R) (Anderson, 2007) provide a well-established framework for modeling fundamental cognitive processes, including decision-making, memory retrieval, and task switching. However, existing literature lacks cognitive models of human responses to interruptions in applied task switching scenarios. One modelling example for interruptions in applied tasks is (Wirzberger & Russwinkel, 2015).

This study investigates the use of cognitive modeling to predict human behavior in an interruption-intensive environment. A controlled cooking task experiment with multiple steps that need to be executed sequentially serves as the foundation of this research. Throughout the task, a robotic assistant provides suggestions. This means that the test subjects must repeatedly interrupt their actual task in order to focus on the robot suggestions, interact with them if necessary and then return to the original task. Thereby, it creates a dynamic environment in which interruptions influence the performance of the task while the usefulness of the robot suggestions variates. To replicate human behavior in this setting, a cognitive model based on ACT-R was developed, incorporating key cognitive mechanisms such as memory retrieval, decision-making, and task resumption.

The study addresses the research question: "How accurately can a cognitive model reproduce human behavior

in an interruption task?" By comparing the model's predictions to human experimental data, this paper evaluates its ability to realistically simulate human task performance in an interruption context. A more precise understanding of the underlying cognitive processes can lead to improvements in human aware AI and human robot interactions, making them more context-aware and less disruptive. Thereby, such an anticipatory robot would have the ability to run a short simulation whether the interruption in this moment would have a disruptive effect or would be supportive. By providing the ability to predict the impact of an interruption on the human partner, this study lays the groundwork for future advancements in intelligent human aware assistance systems that aim to enhance efficiency while minimizing cognitive strain in everyday life.

Background

Multitasking vs. Task Switching

In cognitive psychology, a distinction is made between multitasking and task switching. While both concepts involve managing multiple tasks, they differ in key aspects (Salvucci & Taatgen, 2008; Monsell, 2003). Multitasking refers to performing multiple tasks simultaneously or in rapid succession, often requiring shared cognitive resources. In contrast, task switching describes situations where an individual must fully disengage from one task before resuming another. Österdiekhoff, Heinrich, Russwinkel, and Kopp (2023) developed an agent framework to predict when switching to another task would be beneficial.

Theories

Several theoretical approaches have been proposed to explain how individuals manage interruptions.

Threaded Cognition The Threaded Cognition framework (Salvucci & Taatgen, 2008) provides a model for multitasking, treating tasks as independent cognitive threads that compete for shared resources such as attention, working memory, or motor control and is used for scenarios where individuals perform multiple tasks concurrently, such as driving while talking on the phone. Within ACT-R, this is implemented by allowing multiple task threads to operate in parallel, with a resource allocation mechanism determining which task gains priority when conflicts arise.

The applicability to the present study is limited, since the task requires users to suspend their primary activity in order to address a suggestion, rather than working on both simultaneously.

Memory for Problem States The Memory for Problem States theory (Borst et al., 2010) extends existing cognitive models by introducing the concept of a problem state, a temporary representation in working memory that is essential for solving complex tasks. Within ACT-R, this is implemented through the imaginal module, which holds task-relevant information that can be accessed without delay.

When a secondary task (interruption) requires its own problem state, the system faces a bottleneck: Only one problem state can be actively maintained at a time. Consequently, the primary task's problem state must be stored in the declarative memory (DM), where it begins to decay. Upon resumption of the primary task, retrieving this state incurs additional cognitive costs and can lead to errors if incorrect information is recalled.

Memory for Problem States provides a strong explanation for the effects of interruption duration and task complexity. However, it does not fully account for all cognitive costs associated with task switching (Borst, Taatgen, & Rijn, 2015) and its implementation complexity makes it less suitable for the scope of this study.

Memory for Goals The Memory for Goals theory (Altmann & Trafton, 2002) provides a simpler yet powerful explanation for interruption handling, making it the most appropriate framework for the present study. This model posits that task goals are stored in the DM when an interruption occurs, with their activation levels decaying over time. The longer the interruption, the more difficult it becomes to retrieve the original goal, leading to increased resumption costs.

This theory also accounts for the role of task complexity: during an interruption, individuals rehearse the primary goal to maintain its activation level. When the secondary task is highly demanding, less rehearsal occurs, resulting in greater retrieval difficulty. While some debate exists regarding the extent of rehearsal during interruptions (Katidioti, Borst, & Taatgen, 2014), this framework effectively captures key mechanisms underlying task suspension and resumption. Its ability to explain task-switching costs, particularly in scenarios with critical goal retrieval, makes it well-suited for modeling human behavior in interruption-driven environments.

Methods

To gain further insights into the capabilities of a cognitive model, more specifically ACT-R, regarding interruption behavior, pilot data were collected first in an experiment focusing on an interruption process.

Experiment Setup

The experiment consists of a simple task that simulated a cooking process in which participants are required to follow a four-step recipe. This task was selected in order to have a task that consists of several steps and is based on a realistic application context for cooperation, i.e. working together towards the same goal. Each step consists of an ingredient and a corresponding preparation method which the participants had to select by in-

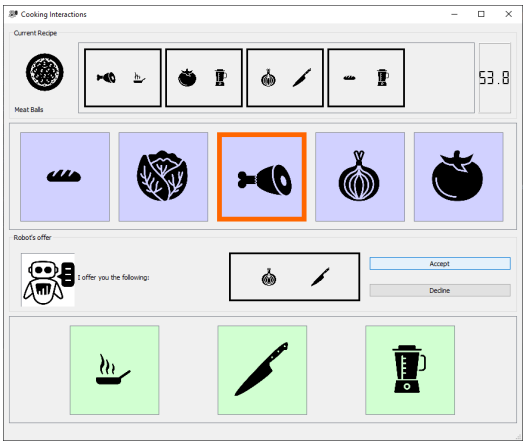


Figure 1: User interface of the experimental design where the participants had to select combinations of ingredients (second row) and preparation methods (bottom) shown in the current recipe (top) while being presented with robot suggestions for such steps that can be declined or accepted (third row)

teracting with a user interface (UI). For each selection made in the UI, a predefined cool down period was implemented to represent the required work time, during which no additional selections could be made. A visual representation of the UI is provided in figure 1. The participants' objective is to complete the recipe as quickly as possible while making as few errors as possible.

During the experiment, a robot assistant provides step suggestions, temporarily interrupting the participant's progress. The participant can either accept the robot's suggestion, allowing the robot to execute the recommended step, decline it, after which a new suggestion is generated following a random delay (max.10 sec), or ignore it. To discourage participants from blindly accepting all suggestions, accepting an incorrect step results in a time penalty. To investigate the impact of different robot suggestions, two types of robot assistants were tested: (1) A random robot that suggests a completely random combination of ingredient and preparation method and (2) a realistic robot which suggests a step, i.e. a combination of ingredient and preparation method, that appears in at least one recipe, ensuring more relevant recommendations. Out of the 15 possible combinations of ingredients and preparation methods, 8 of them occur in at least one of the recipes.

Metrics

To evaluate task performance for both the human participants and the model, completion times at four critical points during the cooking process was analyzed.

1. First interaction with the UI: Time participants spend reading the recipe and planning before starting
2. First interaction with a robot suggestion: Initial reaction time to a suggestion, either accepted or declined
3. Midpoint of the recipe: Two steps completed
4. Recipe completion: Total time taken to finish the task

Data Collection and Model Development

A total of 11 participants took part in the study across two main phases of data collection and model development or testing, respectively. Each participant completed 16 recipes, each recipe twice per robot assistant to ensure consistent data collection.

1.a. Initial Data Collection The initial pilot study was conducted at the Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany with three participants. This preliminary data was used to develop an initial approximation of the cognitive model.

1.b. Model Development The cognitive model was refined in two steps.

(1) Initial Model Testing: Two different cognitive modeling approaches were tested. Each model completed eight recipes per robot to mirror human testing conditions. The models were compared to the initial pilot participant data.

(2) Model Optimization and Reproducibility Testing: ACT-R parameters were adjusted based on initial results to improve accuracy. The refined model performed three full experimental runs to ensure data consistency and detect potential inconsistencies. Through these iterative refinements, the model was optimized to more accurately simulate human behavior in this interruption-prone environment.

2.a. Second Data Collection After refining the model, a second phase of data collection was conducted with eight additional participants at the University of Lübeck, Germany. These participants ranged in age from 22 to 60 years, with a median of 25. The sample included two female and six male participants.

2.b. Model Validation In this final model evaluation, the model was tested against these independent participant data (evaluation participants) to test the models predictions. This evaluation dataset served as an independent test set, ensuring that the model's predictions were not biased by the initial data used to develop the model.

Model

The cognitive model used in this experiment is based on ACT-R (Anderson, 2007)[1]. It is given a virtual window representing the cooking task explained in the experiment setup and a mouse cursor to interact with this window.

[1]Models, data, and experimental setup are available on GitHub `https://github.com/TimoWiesner/predicting_disruptiveness_of_interrupts`

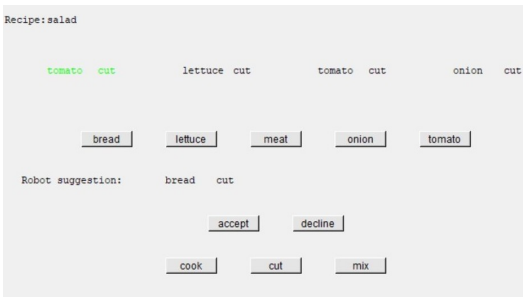


Figure 2: Interface provided to the ACT-R model where the general structure is similar but visually simplified compared to the UI presented to the participants

In relation to the UI in figure 1, the corresponding interface provided to the model can be seen in figure 2.

Model 1

The first model begins by analyzing the recipe and creating an imaginal chunk containing the recipe name and its corresponding steps. This information is then stored in the DM for later recall. Once this initialization is complete, the model starts to execute the cooking task.

Following a left-to-right sequence, the model processes each recipe step by creating imaginal chunks for individual ingredient and preparation selections. Then, it interacts with the UI by clicking the corresponding buttons for each step. Between button clicks, the model continuously checks for robot suggestions. If no suggestion is found, it proceeds with the next step.

When a robot suggestion appears, the model clears its current imaginal chunk and stores a new imaginal chunk containing its current goal buffer values. This ensures that the model can later retrieve its goal status after handling the interruption. This step implements the Memory for Goals theory (Altmann & Trafton, 2002) within the model.

Once the goal buffer is saved in the DM, the model attends the robot's suggestion, generating an imaginal chunk with the suggested ingredient and preparation method. The model then shifts its visual attention to the recipe name and attempts to retrieve the corresponding recipe from the DM. If retrieval is successful, the retrieved recipe steps are compared to the suggested step stored in the imaginal buffer. If a match is found, the suggestion is accepted. Otherwise, it is declined.

After handling the suggestion, the model retrieves the previously stored goal chunk, updates its current goal, and resumes the cooking task from the exact point of interruption, provided that the correct goal chunk is successfully retrieved. This process is illustrated in figure 3 including the gray boxes.

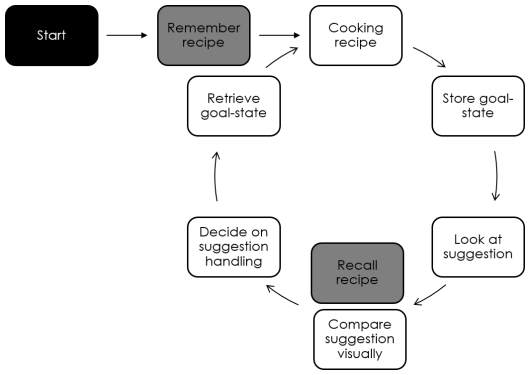


Figure 3: A simplified workflow of the two model approaches is shown. Model 1: including the two gray boxes, i.e. the steps to remember the recipe at the beginning and recall it later, Model 2: only the white boxes, i.e. the recipe is not remembered at the beginning and the robot suggestions are compared visually later on.

Model 2

The second model is similar to the first model except for two key aspects. This process can also be seen in figure 3 whereby the steps shown in the gray boxes are skipped. The first main difference concerns the attention to the detailed steps of the recipe. Instead of storing the complete sequence of steps, the model only memorizes the recipe name and immediately proceeds with the first cooking step. The second difference lies in how the model interacts with the robot's suggestion. Before attending the suggestion, the model saves its current goal into an imaginal chunk, similar to the first model's approach. After storing its goal, the model encodes both the suggested ingredient and the preparation method into the imaginal buffer. However, rather than retrieving the recipe from DM and comparing the suggestion to stored recipe steps, the model directly examines the unfinished steps still visible in the UI. If a matching step is found, the model accepts the suggestion. If no match is found, the model typically ignores the suggestion. Alternatively, it may also decline it when it anticipates that a new suggestion might be more relevant to the recipe. This was implemented in ACT-R by a slightly higher utility for ignoring than declining.

Model parameters The following ACT-R (Anderson, 2007) parameters were used in the second model.

egs 0.5: This is a parameter which adds noise to the utility of the productions to include some randomness.

bll 0.5: This parameter enables base-level learning. A positive value sets the decay parameter and the strongly recommended value of 0.5 was chosen for this model.

ans 0.2: This parameter adds instantaneous noise to the activation, similar to the noise for utilities explained before lying in the recommended range [.2,.8].

Results

To stay within scope, only the final evaluation results are presented. The second modeling approach aligned more closely with human pilot data than the first and was therefore chosen for comparison. Although still slower than humans, it showed improved performance, mainly due to reduced retrieval time during interruptions. Unlike the first model, it retrieved only the goal state – not the entire recipe – leading to more efficient task resumption and closer approximation of human behavior.

Participant Results

One participant was excluded due to completing very few recipes on time, indicating difficulties understanding the task. Additionally, eight more trials (mainly the first and second recipes) were excluded for not being completed within the 1-minute limit, likely due to initial issues with understanding the UI and task.

Comparison

The model data for the comparison is the data from the specified second model approach which was providing the most promising results in the previous test iterations.

Random Robot Looking at figure 4, the data collected in the evaluation tests with the participants matches closely to the initially collected pilot data, i.e. shows similar discrepancy between the model's results and the human data. Especially the first interaction with the robot diverges strongly with the human results with a difference of 7.54 seconds (sec) to the prediction.

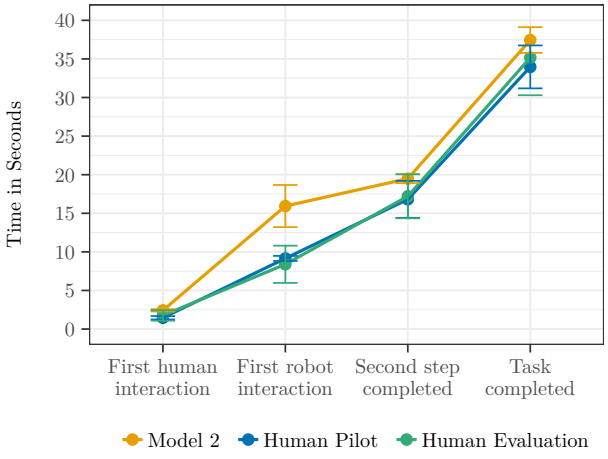


Figure 4: Mean (and SD) time in seconds shown for trials with random robot suggestions for both data sets (pilot and evaluation data) and the data generated by the second model (the one that fitted the pilot data best)

Overall, the data aligns closely with the model's predictions. The initial UI interaction was only 0.3 sec slower, differing by 0.66 sec from the prediction. Task completion averaged 35.16 sec – 2.29 sec faster than predicted. The model showed a lower standard deviation (SD) than participants, except for the first robot suggestion interaction (model: 2.73 sec, participants: 2.41 sec). The model is slower in the first suggestion interaction but reaches halfway through the recipe much faster than participants. From there to completion, the difference is minimal (0.003 sec), indicating similar efficiency in task completion.

Realistic Robot Figure 5 shows that the model's predictions closely align with human data. The initial UI interaction is just 0.6 sec faster, and the largest gap is in the first suggestion interaction (2.97 sec to evaluation data, 3.65 sec to pilot data). The model is closest after completing two steps (1.01 sec difference) and finishes the recipe only 0.53 sec apart. Overall, the model's data shows lower variance than human data.

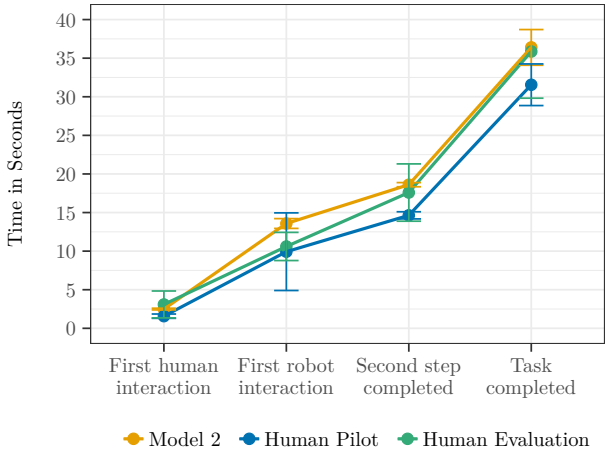


Figure 5: Mean (and SD) time in seconds shown for trials with realistic robot suggestions for both data sets (pilot and evaluation data) and the data generated by the second model (the one that fitted the pilot data best)

The model was slightly faster than participants in its initial UI interaction. It then took 11.08 sec to first interact with a robot suggestion, compared to 7.51 sec for participants – a difference of 3.57 sec. Completing the second step took the model 5.03 sec, 1.96 sec faster than participants (6.99 sec). From there to completion, the model needed 17.79 sec, just 0.49 sec faster than participants (18.27 sec).

Discussion

Accuracy and Predictive Power

The results of this study indicate that the cognitive model successfully approximates human behavior in an interruption-prone task, particularly in terms of task execution time. A comparison of timing data between the

model and human participants reveals minimal deviations with differences of less than one second for task initiation and from step two to recipe completion. This suggests that the model effectively captures sequential task processing and task resumption after interruptions.

The poor performance of the first modeling approach provides insights into human task completion strategies, suggesting that participants did not rely heavily on memorizing the entire recipe before starting. Instead, they appeared to adopt a step-by-step execution approach. This behavior aligns with the improved performance of the second model, which prioritized immediate task execution with minimal retrieval demands.

Additionally, the model's interaction with robot suggestions closely aligns with human data when dealing with realistic robot suggestions. The model also demonstrated a sensitivity to suggestion quality aligning with the initially collected participant data. However, this was not reflected in the evaluation participants. This further supports its validity in simulating human decision-making in interruption scenarios.

However, differences emerged in interactions with the random robot. The model displayed a stronger tendency to ignore irrelevant suggestions for a prolonged period compared to human participants. This discrepancy suggests that the utility parameter for ignoring the robot's suggestions may require adjustment to better align the model's behavior with human responses. Refining this parameter could enhance the model's responsiveness and further improve its similarity to human decision-making.

Limitations and Challenges

One limitation is the lack of a detailed error analysis. Due to the study design, it was difficult to analyze the exact types of errors that were made. However, it might be interesting to have a closer look if the errors made by the model are comparable to the type and amount of errors made by the human participants. Also, the evaluation phase always began with realistic robot suggestions, potentially introducing order effects that influenced participant performance. While the model successfully approximated human behavior, a closer look at the difference of relevance of the interruptions and the cognitive processes and decision times would give a more comprehensive understanding of the theoretical concept behind the results. This study relied on the Memory for Goals theory (Altmann & Trafton, 2002), which provided a strong and interpretable framework for task resumption. However, Memory for Problem States (Borst et al., 2015) may offer a more detailed explanation of cognitive processes during interruptions, especially in complex multitasking scenarios. Future research should explore this alternative framework to assess its potential for even more precise human behavior predictions.

Outlook

Expanding Cognitive Model Applications The findings of this study suggest that cognitive models can simulate human behavior in an interruption context with sufficient accuracy to provide alternative training data, reducing reliance on direct human testing. Additionally, a well fitted cognitive model could provide insight into underlying processes such as memory retrieval times in different stages of task execution and therefore help to improve human-robot interactions by more accurately anticipating the human's behavior. However, some trade-offs in accuracy remain.

While cognitive models cannot fully replace human data collection, they can significantly reduce data acquisition costs while maintaining reasonable fidelity. This is particularly valuable for human-centered AI systems, where understanding and predicting user behavior is crucial for designing adaptive, interactive AI assistants.

Enhancing Model Variability To improve the usefulness of model-generated data, future studies should focus on increasing model variability. A promising approach is to run the model with different cognitive parameters, simulating variations in human behavior due to factors such as age, cognitive flexibility, or experience levels. This would enhance the dataset's richness, making it more applicable for training robust machine learning models.

Validating ACT-R as a Quantitative Data Source This study provides further validation of ACT-R as a framework for simulating human cognition in task-switching environments. While ACT-R has long been used for theoretical cognitive modeling, its potential as a quantitative data generator remains an emerging area of research. The results support the feasibility of using ACT-R for structured data collection, but further cross-validation with real-world experiments is necessary to establish its reliability for AI development.

Conclusion This paper shows that cognitive models can feasibly predict behavior in interruption-prone tasks and serve as alternative AI training data. However, task specificity and small sample size limit generalizability.

Looking ahead, integrating more diverse cognitive parameters, exploring alternative theoretical models, and conducting larger-scale controlled studies will be critical steps in refining cognitive models as scalable tools for human-centered AI research. As AI technology continues to evolve, leveraging cognitive models in this way could significantly advance AI's ability to interact intelligently and seamlessly with human users.

Acknowledgments

We would like to thank Adna Bliek for valuable discussions and insights that helped shape the ideas presented in this paper.

References

Adamczyk, P., & Bailey, B. (2004). If not now, when?: The effects of interruption at different moments within task execution. In (Vol. 6, pp. 271–278).

Altmann, E., & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive Science*, *26*, 39–83.

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

Bitkina, O., Kim, J., Park, J., Park, J., & Kim, H. K. (2021). User stress in artificial intelligence: Modeling in case of system failure. *IEEE Access*, *PP*, 1–1.

Borst, J., Taatgen, N., & Rijn, H. (2010). The problem state: A cognitive bottleneck in multitasking. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *36*, 363–82.

Borst, J., Taatgen, N., & Rijn, H. (2015). What makes interruptions disruptive? In (pp. 2971–2980).

Katidioti, I., Borst, J. P., & Taatgen, N. A. (2014). What happens when we switch tasks: Pupil dilation in multitasking. *Journal of Experimental Psychology: Applied*, *20*(4), 380–396.

Koundal, N., Abdalhadi, A., Al-Quraishi, M., Elamvazuthi, I., Moosavi, M., Chr, G., ... Saad, N. (2024). Effect of interruptions and cognitive demand on mental workload: A critical review. *IEEE Access*, *PP*, 1–1.

Monsell, S. (2003). Task switching. *Trends in Cognitive Sciences*, *7*, 134–140.

Österdiekhoff, A., Heinrich, N. W., Russwinkel, N., & Kopp, S. (2023). Sense of control in dynamic multitasking and its impact on voluntary task-switching behavior. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 46).

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review*, *115*(1), 101–130.

Speier, C., Vessey, I., & Valacich, J. (2003). The effects of interruptions, task complexity, and information presentation on computer-supported decision-making performance. *Decision Sciences*, *34*.

Wirzberger, M., & Russwinkel, N. (2015). Modeling interruption and resumption in a smartphone task: An act-r approach. *i-com*, *14*, 147–154.