Holographic Declarative Memory: Using Distributional Semantics within ACT-R

Matthew A. Kelly (matthew.kelly@psu.edu), David Reitter (reitter@psu.edu)
The Pennsylvania State University, University Park, PA

Abstract

We explore replacing the declarative memory system of the ACT-R cognitive architecture with a distributional semantics model. ACT-R is a widely used cognitive architecture, but scales poorly to big data applications and lacks a robust model for learning association strengths between stimuli. Distributional semantics models can process millions of data points to infer semantic similarities from language data or to infer product recommendations from patterns of user preferences. We demonstrate that a distributional semantics model can account for the primacy and recency effects in free recall, the fan effect in recognition, and human performance on iterated decisions with initially unknown payoffs. The model we propose provides a flexible, scalable alternative to ACT-R's declarative memory at a level of description that bridges symbolic, quantum, and neural models of cognition.

Introduction

The ACT-R cognitive architecture (Anderson & Lebiere 1998) provides a symbolic, high-level account of declarative memory that has explained much behavioral data. Distributional models of memory (e.g., Jones & Mewhort 2007) can describe exactly *how* memories and associations are formed and work well in big-data contexts (Rutledge-Taylor, Vellino, & West 2008). Are symbolic and distributional architectures compatible, and can they be unified?

We demonstrate that a distributional model integrated with ACT-R can effectively substitute for the ACT-R Declarative Memory (DM). Our model, Holographic Declarative Memory (HDM), accounts for primacy and recency effects in free recall, the fan effect in recognition, and human performance on iterated decision. HDM provides a flexible, scalable alternative to ACT-R's DM. HDM is a vector-symbolic (Gayler 2003) or conceptual space (Lieto, Chella, & Frixione 2017) model, and as such, operates at a level of description that is the *lingua franca* of cognitive modeling, bridging symbolic, quantum, and neural models. In developing HDM as a module for ACT-R, our intent is help advance cognitive science toward a cognitive architecture capable of modeling human performance at all scales of learning.

ACT-R

The ACT-R cognitive architecture consists of perceptual and motor modules that interact with the agent's environment, working memory buffers which hold the active data in the agent's mind, a declarative memory that holds the agent's world knowledge, and a procedural memory that controls the flow of information and evaluates possible actions.

ACT-R's declarative memory (DM) consists of items of knowledge, called *chunks*, weighted by an estimate of the probability that chunk is useful in the agent's current context. Each chunk is a list of *slot:value* pairs (e.g., "name:tiger type:animal has:stripes").

In ACT-R, the agent's current context serves as a cue to the memory system. Chunks in memory are activated according to a combination of the chunk's base-level activation, which reflects how frequently and recently that chunk has been accessed, and the amount of activation that spreads from the cue to the chunk, which reflects the strength of the association between the cue and the chunk. More active memories are retrieved more easily and quickly.

In vector-symbolic models, each memory is represented by a vector, which can be understood as a point on the surface of a hypersphere of all possible memories. The distance from the cue's vector to a given memory's vector in this high-dimensional space is the activation of that memory with respect to the cue. The geometries of the space estimate the probability of the relevance of the memory given the cue (Kelly, Kwok, & West 2015). As such, the distance can be understood as a realization of DM's spreading activation.

BEAGLE

In the BEAGLE model of semantic memory (Jones & Mewhort 2007), each word is represented by two vectors: an environment vector that represents the percept of a word and a memory vector that represents the concept of a word.

An environment vector (denoted by **e**) stands for what a word looks like in writing or sounds like when spoken. In our simulations, environment vectors are generated by randomly sampling values from a Gaussian distribution with a mean of zero and a variance of $1/n$, where n is the dimensionality. In BEAGLE, the dimensions are meaningless, only the relationships between vectors are meaningful. The number of dimensions, n, determines the fidelity with which BEAGLE stores the word co-occurrence information from a corpus, such that smaller n yields poorer encoding.

Memory vectors (denoted by **m**) represent the associations a word has with other words. Memory vectors are con-

structed as the model reads the corpus. Memory vectors are holographic in that they use circular convolution (denoted by $*$) to compactly encode associations between words (Plate 1995). Given a sentence, for each word in the sentence, vectors representing all sequences of words in the sentence (or grams) that include the target word are summed together and added to the target word's memory vector.

When adding grams to a target word's memory vector, each word in those grams is represented by an environment vector, with the exception of the target word, which is replaced by the placeholder vector. The placeholder vector is generated randomly like an environment vector. We use "?" to denote the placeholder, as it functions much like a question mark. The memory vector for a word can be understood as the sum of all questions to which that word is the answer.

Given the sentence, "eagles soar over trees", we update the memory vectors for each word in the sentence: $\mathbf{m}_{\text{eagles}}$, $\mathbf{m}_{\text{soar}}$, $\mathbf{m}_{\text{over}}$ and $\mathbf{m}_{\text{trees}}$. Each memory vector is updated with a sum of grams. $\mathbf{m}_{\text{soar}}$, is updated with the bigrams "eagles ?" and "? over", the trigrams "eagles ? over" and "? over trees", and the tetragram "eagles ? over trees".

BEAGLE's algorithm is not specific to language and has been applied to iterated decision, math cognition, and playing simple games (Rutledge-Taylor et al. 2014). BEAGLE can also be used as a recommender-system for films and research papers (Rutledge-Taylor, Vellino, & West 2008).

Holographic Declarative Memory

Holographic Declarative Memory (HDM), first proposed by Kelly, Kwok, & West (2015), is a module for the Python implementation of ACT-R (Stewart & West 2007). Both Python ACT-R and HDM are available through GitHub[1].

HDM is a variant of BEAGLE designed to interface with the ACT-R procedural memory. ACT-R can add chunks to and request chunks from HDM. However, unlike DM, HDM does not store chunks. Instead, HDM stores associations between *values*. Each value is represented in memory by a holographic vector that moves in relation to the vectors of other values as chunks are added to HDM. Next, we review how cognitive operations are performed, from elementary ones to complex ones for which empirical data exist.

Add a chunk with slots

Typically in ACT-R, a chunk consists of an unordered set of *slot:value* pairs. HDM represents each *slot* by a randomly generated permutation $\mathbf{P}_{\text{slot}}$. Each *value* is represented by an environment vector $\mathbf{e}_{\text{value}}$ and a memory vector $\mathbf{m}_{\text{value}}$.

Each $\mathbf{m}_{\text{value}}$ is a sum of questions to which the value is an answer. E.g., when the chunk "color:red shape:square size:large" is added, $\mathbf{m}_{\text{red}}$, $\mathbf{m}_{\text{square}}$, and $\mathbf{m}_{\text{large}}$ are updated. To update $\mathbf{m}_{\text{red}}$, the four questions "What color is it?", "What color is the square?", "What color is the large thing?" and "What color is the large square?" are added:

$$\begin{aligned}\mathbf{m}_{\text{red},t} = \alpha\mathbf{m}_{\text{red},t-1} &+ \mathbf{q}_{\text{color:?}} + \mathbf{q}_{\text{color:? shape:square}} \\ &+ \mathbf{q}_{\text{color:?}} + \mathbf{q}_{\text{color:? size:large}} \\ &+ \mathbf{q}_{\text{color:? shape:square size:large}}\end{aligned} \quad (1)$$

where *t* is the current time step and α is the forgetting rate. Each question is represented by a cue vector, $\mathbf{q}$, constructed by permuting each $\mathbf{e}_{\text{value}}$ by the corresponding $\mathbf{P}_{\text{slot}}$ and then convolving. Using the placeholder vector, $\mathbf{\Phi}$, the question "What color is the large square?" is constructed as:

$$\mathbf{q}_{\text{color :? shape:square}} = (\mathbf{P}_{\text{color}}\mathbf{\Phi}) * (\mathbf{P}_{\text{shape}}\mathbf{e}_{\text{square}}) \quad (2)$$

Add a chunk without slots

Python ACT-R allows chunks that are ordered sequences of values (e.g., "large red square"). To encode chunks without slots, HDM uses the permutation $\mathbf{P}_{\text{before}}$ recursively to created nested permutations. The cue vector for "What came after *large* and before *square*?" or "large ? square" is:

$$\mathbf{q}_{\text{large ? square}} = (\mathbf{P}_{\text{before}}((\mathbf{P}_{\text{before}}\mathbf{e}_{\text{large}}) * \mathbf{\Phi})) * \mathbf{e}_{\text{square}} \quad (3)$$

HDM uses skip-grams such that values in a chunk do not need to be consecutive to be associated with each other. For example, in "large red square", "? square" (i.e., "What came before square?") is added to both $\mathbf{m}_{\text{large}}$ and $\mathbf{m}_{\text{red}}$.

Recall: request with one unknown

To recall something, the request to HDM must provide a chunk with exactly one unknown (e.g., "color:? shape:square size:large"). In *exact matching*, the chunk is represented by the corresponding cue vector. In *partial matching*, the chunk is decomposed into all sub-questions (e.g., "What color is it?", "What color is the square?", "What color is the large thing?", and "What color is the large square?") and represented as the sum of those questions.

The memory vector with the highest similarity to the cue vector, or sum of cue vectors, is selected as the response. Similarity between vectors is measured by the vector cosine. The unknown in the chunk is then replaced with the response and the modified chunk is returned.

Recognition: request with no unknowns

To recognize something, the request to HDM must provide a chunk with no unknowns. HDM then computes the *coherence* of the chunk. *Coherence* is calculated as the mean cosine between the memory vector for each value in the chunk and the cue vectors for a chunk with that value substituted for an unknown, e.g., for "color:red shape:square":

$$\begin{aligned}0.5(&\text{cosine}(\mathbf{q}_{\text{color:?}} + \mathbf{q}_{\text{color:? shape:square}}, \mathbf{m}_{\text{red}}) \\ &+ \text{cosine}(\mathbf{q}_{\text{shape:?}} + \mathbf{q}_{\text{color:red shape:?}}, \mathbf{m}_{\text{square}}))\end{aligned} \quad (4)$$

Decay and the Serial Position Curve

In DM, each chunk i in memory has a base level activation B_i that decays as a power function of the time since the chunk was last added to memory,

[1]Python ACT-R with HDM can be downloaded from `https://github.com/MatthewAKelly/ccmsuite` and example ACT-R HDM models can be downloaded from `https://github.com/MatthewAKelly/HDM`

$$B_i = \ln(\sum_{j=1}^{n} t_j^{-d}) \quad (5)$$

where n is the number of times chunk *i* is presented, t_j is the time since the *j*th presentation, and d is the rate of decay.

Conversely, HDM has association strengths, but no base level activation. The activation of a value is a function of the distance on the surface of the hypersphere between that value and a given cue. The structure of HDM commits us to representing the availability of information in memory as entirely contingent on associations.

The serial position effect (Ebbinghaus 1885) is the finding that when people study a list, the items at the beginning and end of the list are remembered best. The recall advantage for items at the beginning of the list (the *primacy effect*) is generally attributed to participants having longer to process and rehearse those items. There are, however, several competing theories of the recall advantage for items at the end of list (the *recency effect*). In DM, the recency effect is due to decay. In distributed processing models, a recency effect can be modeled as interference.

In models such as holographic memories and neural networks, whenever an item is added to memory, the memory engram for that item interferes with the engrams of prior items (retroactive interference), and is also interfered with by those prior engrams (proactive interference).

TODAM, a holographic memory model (Murdock 1982), uses a forgetting parameter α to update memory,

$$\mathbf{m}_i = \alpha \mathbf{m}_{i-1} + \mathbf{v}_i \quad (6)$$

where $\mathbf{v}_i$ is the vector for a memory engram and $\mathbf{m}_{i-1}$ and $\mathbf{m}_i$ are the memory vector before and after storage. The forgetting parameter α ranges from 0 to 1. Multiplying the memory store by α privileges new information over old information, allowing retroactive interference to be stronger than proactive interference, which produces a recency effect. We use α as a parameter to control forgetting in HDM. Franklin & Mewhort's (2015) holographic model accounts for both primacy and recency effects in terms of rehearsal and interference without needing a time-based memory decay function. Franklin and Mewhort's model is a single holographic memory vector that stores all list items. As such, all new items stored interfere with all previous items.

Conversely, HDM has one memory vector per item, such that interference occurs only between different associations for a given item. As a result, there is less interference in HDM than in a single vector model. This property allows HDM to better handle big data, such as modeling language learning, but at the cost of making HDM a less accurate model of small scale memory tasks such as list learning.

To compensate, we add a time-based decay function to HDM. We hold that this time-based decay function is properly understood as a proxy for sources of interference that have not been explicitly modeled. Decay is implemented by adding noise over time to all memory vectors,

$$\mathbf{m}_t = \mathbf{m}_{t-1} + \eta \mathbf{n} \quad (7)$$

where $\mathbf{m}$ is a memory vector, t is the time in seconds, $\mathbf{n}$ is a random vector, and η is the noise parameter. If η is zero, no noise is added and there is no decay. For larger η, more noise is added per second and decay is steeper.

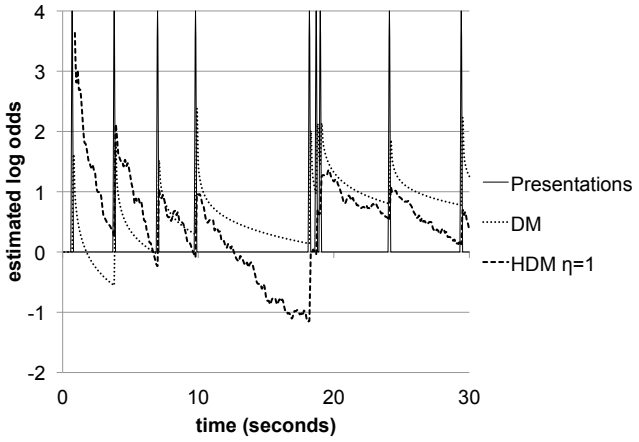

Figure 1: Activation of a chunk over time in DM and HDM.

To compare DM and HDM, we treat the *coherence* of a chunk in HDM as analogous to base level activation in DM. In Figure 1, we compare activation of a chunk over 30 seconds in DM and HDM without parameter fitting. DM has $d = 0.5$ and HDM has dimensionality = 64, $\alpha = 0.9$, and $\eta = 1$. The chunk is repeatedly stored in memory at random intervals indicated in Figure 1 by vertical lines.

Activation in DM is an estimate of the log odds of the relevance of the chunk to the current situation. Similarly, cosine in HDM is an estimate of the root probability of the relevance of the chunk. Accordingly, for Figure 1, we convert cosine *C* to activation *A* as follows:

$$A = \ln(\frac{C^2}{1 - C^2}) \quad (8)$$

We find that adding noise in HDM is more stochastic than decay in DM but produces comparable performance.

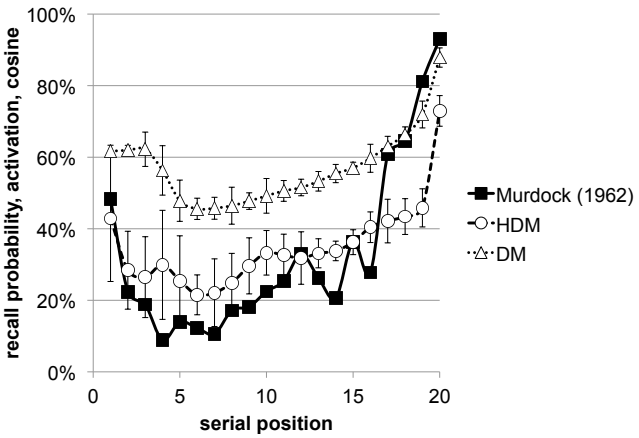

Figure 2: Performance as a function of list position.

In Figure 2 we compare two ACT-R models of the serial position effect to human data from Murdock (). The

two models are identical except that one uses DM and the other uses HDM. Participants and models were presented 20 words at a rate of one word every 2 seconds. After, participants reported back the list in any order (free recall). For simplicity, the models did not report back the list and instead we use the state of memory as a proxy for recall probability.

To study the list, the models stores chunks that contain pairs of items: the current item and the previous item. There are 22 items in total: the 20 words of the list, the start list cue, and the end list cue. There are 21 chunks, one for each pair of items: (*START 1*, *1 2*, ... *9 10*, *10 END*). In Figure 2, the activation of a given item is calculated as the mean of the activations of the two chunks that the item appears in (e.g., for word *6*, the activation is the mean of the chunks *5 6* and *6 7*). We divide DM activation by 4 to convert to predicted recall probability. Error bars indicate standard deviation.

To capture the primacy effect, the models used the following rehearsal strategy: rehearse the chunk for the current and previous item of the list, then return to the start of the list and rehearse forward as far as can be recalled.

DM and HDM use the same equation to determine time to recall a chunk. Retrieval time *RT* is an exponential function of A_i, the cosine or activation of chunk *i*,

$$RT = Fe^{-A_i} \tag{9}$$

where *F* is the latency factor. Because HDM's cosine and DM's activation have different ranges, HDM and DM may need to use different *F*. In a task where DM chunks have $A_i > 1$, retrieval will be faster than for DM than HDM.

For the free recall task, to prevent the models from rehearsing through the entire 20 item list every 2 seconds during the study phase, we used long retrieval latencies for both models. For DM, we used the standard decay rate of $d = 0.5$ and a latency of $F = 3.0$. For HDM, we used a dimensionality of 64, $\alpha = 0.9$, $\eta = 1$, and $F = 0.5$.

Both models strongly correlate with the human data ($r = 0.87$ for DM, $r = 0.90$ for HDM). Parameters were set by hand. Better fits for both models may be obtainable using systematic parameter fitting, but the results in Figure 2 are sufficient to demonstrate that HDM provides at least as good a fit to human forgetting data as DM.

Interference and the Fan Effect

In the fan effect task (Anderson 1974), participants study words pairs, such as person-location pairs (e.g., *hippy-park* or *lawyer-bank*). At test, participants are presented pairs that are either studied (targets, e.g., *hippy-park*) or novel (foils, e.g., *lawyer-park*) and must quickly discriminate.

The *fan* of a word is the number of pairs in the study set that contain that word. For example, if there are three pairs in the study set that contain *hippy* (*hippy-park*, *hippy-bank*, and *hippy-store*), then hippy has a fan of three. The *fan effect* is the finding that, at test, participants are slower to make judgments about words with a higher fan. If *lawyer* has a fan of 1 (i.e., is only in the pair *lawyer-bank*), then participants are faster to make judgments about pairs that contain *lawyer* than they are about pairs that contain *hippy*.

The fan effect illustrates a fundamental principle of human memory: the availability of information in memory is an estimate of the probability that the information is useful in the current situation. If a participant has studied 3 pairs with the word *hippy*, each pair has only a 1 in 3 chance of being useful for judging a test pair that contains *hippy*. Conversely, if the participant has studied only one pair with the word *lawyer*, that pair has a 100% chance of being useful for judging a test pair that contains *lawyer*.

DM models the fan effect by setting the association strengths between the words in the cue and the pairs in memory to a function of the fan of each word (Anderson & Reder 1999). The DM model produces a response time *T* in seconds that is a function of the fans f_{person} and f_{place},

$$T = 0.233(f_{person} f_{place})^{1/3} + 0.845 \tag{10}$$

which correlates well with humans ($r = 0.95$, see Figure 3).

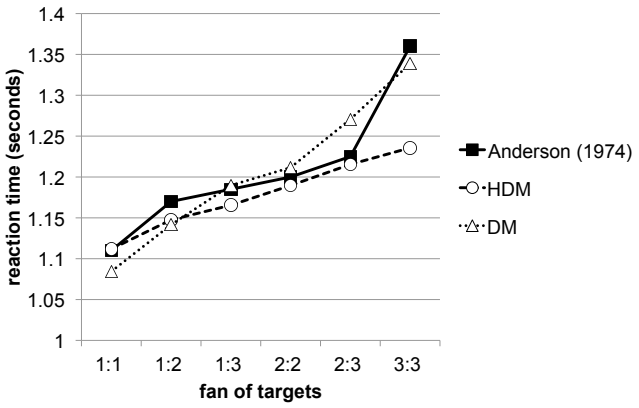

Figure 3: Response time for targets in the fan effect task.

Unfortunately, to build the DM model, associations strengths must be set by hand. Conversely, HDM learns association strengths from experience.

Kelly, Kwok, & West (2015) model the fan effect task using HDM. Without changing *any* parameters in Anderson & Reder's (1999) model of the fan effect, the HDM model provides a good fit to the data ($r = 0.91$). Both the DM and HDM models used a latency of $F = 0.63$ and no decay.

Kelly, Kwok, & West show how the fan effect arises from the geometry of the vector space. The HDM fan effect model uses a dimensionality of 256, but the effect can be illustrated in three dimensions (see Figure 4).

The memory vector for *hippy*, $\mathbf{m}_{\text{hippy}}$, is constructed as a sum of cues. For a fan of 2, those cues are "Who is in the park?" and "Who is in the bank?", respectively represented by the chunks "? park" and "? bank" and the corresponding vectors $\mathbf{q}_{\text{? park}}$ and $\mathbf{q}_{\text{? bank}}$. If these two questions are weighted equally, $\mathbf{m}_{\text{hippy}}$ will be equidistant from $\mathbf{q}_{\text{? park}}$ and $\mathbf{q}_{\text{? bank}}$. HDM uses randomly generated vectors that are orthogonal in expectation. If we assume the vectors are perfectly orthogonal, $\mathbf{m}_{\text{hippy}}$ will be at a 45°angle from $\mathbf{q}_{\text{? park}}$ and $\mathbf{q}_{\text{? bank}}$ with a cosine of 0.71.

At a fan of 3, $\mathbf{m}_{\text{hippy}}$ is equidistant from $\mathbf{q}_{\text{? park}}$, $\mathbf{q}_{\text{? bank}}$ and $\mathbf{q}_{\text{? store}}$ with a 55°angle and a cosine of 0.58.

As the fan increases, the angle between memory and cues increases. For a fan of f and perfectly orthogonal vectors,

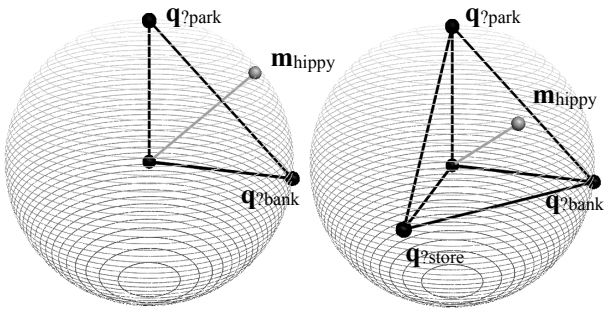

Figure 4: $\mathbf{m}_{\text{hippy}}$ with a fan of 2 (left) or 3 (right). Figure from Kelly, Kwok, & West (2015).

the cosine is $f^{-1/2}$, i.e., the square root of the probability of the item conditional on the cue (Kelly, Kwok, & West 2015).

In the fan effect task, all pairs of words in the study set are equiprobable. For events with unequal probabilities, the cosine underestimates the probability of low frequency events. Kelly, Kwok, & West (2015) note that for *n* events with frequencies v_1 to v_n, the cosine of event *i* is:

$$\text{cosine} = \frac{v_i}{\sqrt{v_1^2 + ... + v_i^2 + ... + v_n^2}} \tag{11}$$

HDM can be understood as a realization of the quantum probability model of human judgment within a cognitive architecture. Like HDM, quantum probability models (Busemeyer et al. 2011) use the cosine between vectors to model human probability judgments.

However, HDM differs from quantum probability models in two important ways: (1) HDM can learn the strengths of associations between items from experience whereas in quantum models the vector similarities are set by hand, and (2) quantum probability models use the square root of the frequencies and the square of the cosine so as to compute the exact probability. HDM is unable to do (2) because of (1). Taking the square root of the frequencies requires knowing the frequencies *a priori*, whereas HDM learns the frequencies gradually through experience.

Like HDM, when making decisions from experience, people tend to underestimate the probability of rare events (Hertwig et al. 2004). The cosine's biased estimate of probability may support HDM's validity as a cognitive model.

Procedural Learning and Decision Making

Procedural and declarative memory are often characterized as memory of *how* and *what*, respectively. Procedural memory consists of "if *condition* then *action*" production rules weighted by utilities that estimate how good it is to do *action*. Declarative memory consists of information weighted by how useful it is to remember that information given a cue.

At a high level of description these two systems are the same: "if *cue* then *information*" is not much different from "if *condition* then *action*". Chunks and production rules are both weighted by probability estimates. Activation estimates

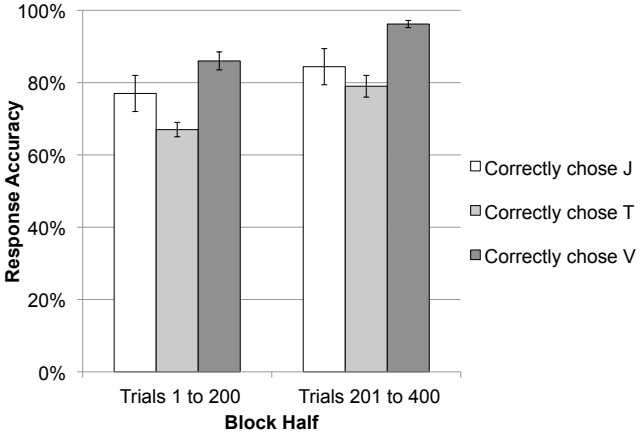

Figure 5: Rate at which participants makes optimal decisions. Figure adapted from Walsh & Anderson (2011).

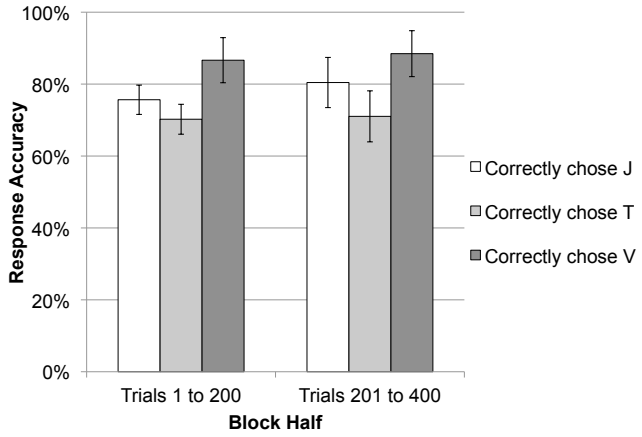

Figure 6: Rate at which HDM makes optimal decisions.

the probability that a chunk is useful to know and utility estimates the probability that a production rule is useful to do.

The usefulness of *knowing* and *doing* are distinct. For example, *knowing* that touching a sharp object will hurt you is useful. *Touching* a sharp object is not. Nevertheless, the functional similarity between procedural and declarative memory suggest that the same model of memory could be used to implement both systems.

Walsh & Anderson (2011) have human participants perform an iterated binary decision task with initially unknown payoffs. The tasks consists of a choice, made by pressing one of two keys, followed by an abstract cue, and then a second choice. After the second choice, participants receive either positive or negative feedback. This completes a single trial of the task. The probability of positive feedback is contingent on the first and second choice as well as the cue. The task is difficult to learn as optimal choices yield a maximum of a 50% chance of positive feedback.

As shown in Figure 5, over 400 trials, participants gradually learn to perform the task well. Results are from Walsh

& Anderson (2011). Data is averaged over 26 participants. Error bars indicate standard error.

Kelly & West (2013) apply HDM to Walsh & Anderson's task. HDM is initialized to a state of optimism. Each possible decision in the task, $decision_i$, is initially associated with positive feedback, *good*, by adding the chunk "$decision_i$ *good*" to memory 30 times.

Kelly & West find that optimism motivates the model to explore the decision space. This is consistent with the *broaden-and-build* (Fredrickson 2001) theory of positive emotions, which holds that positive emotions broaden the repertoire of actions considered when making decisions.

Each completed trial is represented sequentially as a chunk of the form "*start decision1 cue decision2 feedback*" and added to memory. The first decision is made by querying memory with "*start ? good*" and the second is made by querying with "*start decision1 cue ? good*". The HDM model learns to perform the task well at a rate similar to humans (see Figure 6). The HDM model uses 256 dimensional vectors. Results are averaged across 26 runs of the model.

A critical difference between procedural memory and declarative memory is that procedural memory uses reinforcement learning. In reinforcement learning, an association is learned as a function of how surprising it is, i.e., the magnitude of the difference between prediction and observation. Walsh & Anderson find that human performance on the task is consistent with temporal difference reinforcement learning models. We speculate that performance of the Kelly, Kwok, & West model could be improved by implementing surprise-driven reinforcement learning in HDM.

Conclusion

In sum, we believe that an integrated theory of cognition should have both a symbolic (e.g., a description in terms of features and values) and a sub-symbolic (e.g., a description in terms of vector algebra) component to provide satisfying explanations, and that distributed semantic representations are a natural candidate for an account of declarative memory and its learning processes.

Acknowledgements

The authors gratefully acknowledge funding from NSF grants SES-1528409 and BCS-1734304.

References

Anderson, J. R., and Lebiere, C. 1998. *The Atomic Components of Thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., and Reder, L. M. 1999. The fan effect: New results and new theories. *Journal of Experimental Psychology: General* 128:186–197.

Anderson, J. R. 1974. Retrieval of propositional information from long-term memory. *Cognitive Psychology* 6:451–474.

Busemeyer, J. R.; Pothos, E. M.; Franco, R.; and Trueblood, J. 2011. A quantum theoretical explanation for probability judgement errors. *Psychological Review* 118:193–218.

Ebbinghaus, H. 1885. *Memory: A Contribution to Experimental Psychology*. New York, N.Y.: Dover.

Franklin, D. R. J., and Mewhort, D. J. K. 2015. Memory as a hologram: An analysis of learning and recall. *Canadian Journal of Experimental Psychology* 69:115–135.

Fredrickson, B. L. 2001. The role of positive emotions in positive psychology: The broaden-and-build theory of positive psychology. *American Psychologist* 56:218–226.

Gayler, R. W. 2003. Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience. In *Proceedings of the Joint International Conference on Cognitive Science*. Sydney, Australia: University of New South Wales. 133–138.

Hertwig, R.; Barron, G.; Weber, E. U.; and Erev, I. 2004. Decisions from experience and the effect of rare events in risky choice. *Psychological Science* 15(8):534–539. PMID: 15270998.

Jones, M. N., and Mewhort, D. J. K. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review* 114:1–37.

Kelly, M. A., and West, R. L. 2013. Decision-making in a dynamically structured holographic memory: Learning from delayed feedback. In West, R., and Stewart, T., eds., *Proceedings of the 12th International Conference on Cognitive Modeling*. Ottawa, Canada: Carleton University. 47–52.

Kelly, M. A.; Kwok, K.; and West, R. L. 2015. Holographic declarative memory and the fan effect: A test case for a new memory model for act-r. In Taatgen, N. A.; van Vugt, M. K.; Borst, J. P.; and Mehlhorn, K., eds., *Proceedings of the 13th International Conference on Cognitive Modeling*. Groningen, the Netherlands: University of Groningen. 148–153.

Lieto, A.; Chella, A.; and Frixione, M. 2017. Conceptual spaces for cognitive architectures: A lingua franca for different levels of representation. *Biol. Inspired Cognitive Architectures* 19:1–9.

Murdock, B. B. The serial position effect of free recall.

Murdock, B. B. 1982. A theory for the storage and retrieval of item and associative information. *Psychological Review* 89:609–626.

Plate, T. A. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks* 6:623–641.

Rutledge-Taylor, M. F.; Kelly, M. A.; West, R. L.; and Pyke, A. A. 2014. Dynamically structured holographic memory. *Biologically Inspired Cognitive Architectures* 9:9–32.

Rutledge-Taylor, M. F.; Vellino, A.; and West, R. L. 2008. A holographic associative memory recommender system. In *Proceedings of the 3rd International Conference on Digital Information Management*, 87–92.

Stewart, T. C., and West, R. L. 2007. Deconstructing and reconstructing ACT-R: Exploring the architectural space. *Cognitive Systems Research* 8(3):227–236.

Walsh, M. M., and Anderson, J. R. 2011. Learning from delayed feedback: Neural responses in temporal credit assignment. *Cognitive, Affective, and Behavioral Neuroscience* 11:131–143.