RUNNING HEAD: DYNAMICALLY STRUCTURED HOLOGRAPHIC MEMORY

Dynamically Structured Holographic Memory

Matthew F. Rutledge-Taylor (m.rutledge-taylor@cogniva.ca)

Cogniva Information Science Research Institute

Gatineau, Quebec, Canada

Matthew A. Kelly (matthew.kelly2@carleton.ca)

Robert L. West (robert_west@carleton.ca)

Institute of Cognitive Science, Carleton University

Ottawa, Ontario, Canada

Aryn A. Pyke (apyke@andrew.cmu.edu)

Department of Psychology, Carnegie Mellon University

Pittsburgh, Pennsylvania, United States of America

Send correspondence to:

Matthew A. Kelly

Institute of Cognitive Science

Carleton University

Ottawa, ON, Canada

Email: matthew.kelly2@carleton.ca

Abstract

We describe the DSHM (Dynamically Structured Holographic Memory) model of human memory, which uses high dimensional vectors to represent items in memory. The complexity and intelligence of human behavior can be attributed, in part, to our ability to utilize vast knowledge acquired over a lifetime of experience with our environment. Thus models of memory, particularly models that can scale up to lifetime learning, are critical to modeling human intelligence. DHSM is based on the BEAGLE model of language acquisition (Jones and Mewhort, 2007) and extends this type of model to general memory phenomena. We demonstrate that DHSM can model a wide variety of human memory effects. Specifically, we model the fan effect, the problem size effect (from math cognition), dynamic game playing (detecting sequential dependencies from memories of past moves), and time delay learning (using an instance based approach). This work suggests that DSHM is suitable as a basis for learning both over the short-term and over the lifetime of the agent, and as a basis for both procedural and declarative memory. We argue that cognition needs to be understood at both the symbolic and sub-symbolic levels, and demonstrate that DSHM intrinsically operates at both of these levels of description. In order to situate DSHM in a familiar context, we discuss the relationship between DHSM and ACT-R.

Dynamically Structured Holographic Memory

Cognitive science, as a discipline, provides explanations for why cognitive phenomena occur and how they occur. The explanation for how a phenomenon occurs often involves a description of what processes underlie it. This need for process level accounts makes modeling, which is explicit in mechanical details, a particularly useful tool in generating explanations for cognitive phenomena.

To achieve a full, theoretical understanding of a cognitive process, explanations need to be provided at both symbolic (i.e., representational) and sub-symbolic levels of description. The classic symbolic approaches to modeling do not account for how the symbol manipulations described in the model could arise from neural tissue, nor do they account for how the symbols themselves come into existence. Classic connectionist approaches are more concerned with neural plausibility, but are notoriously opaque, doing little to aid our understanding of the cognitive processes modeled. By contrast, the vector-symbolic approach to modeling explicitly provides an account at both levels of description.

Vector Symbolic Architectures (VSAs), a term coined by Gayler (2003), are a set of techniques for instantiating and manipulating symbolic structures in distributed representations. Research into VSAs has been motivated by limitations in the ability of traditional connectionist models (i.e., non-recurrent models with one or two layers of connections) to represent knowledge with complicated structure (Plate, 1995). Like human memory, vector symbolic architectures can store complicated and recursive relations between ideas. VSAs use vectors with hundreds of dimensions, but the number of dimensions does not grow with either the quantity or complexity of the experiences stored within the vectors. For a formal analysis of VSAs, including time complexity considerations, see Kelly, Blostein, and Mewhort, 2013 as well as Plate, 1995.

VSAs have been used to successfully model a number of different cognitive processes including analogical mapping (Eliasmith & Thagard, 2001), letter position coding (Hannagan, Dupoux, & Christophe, 2011), semantic memory (Jones & Mewhort, 2007); and working memory (Eliasmith 2013; Franklin & Mewhort, 2013; Murdock, 1982; 1993). Eliasmith (2013) has shown that the linear algebra operations that define VSAs have a natural, efficient implementation in layers of neural connectivity, such that VSAs can be used in realistic models of neural processing.

The aim of this paper is to present Dynamically Structured Holographic Memory (DSHM), a system for modeling human memory that uses holographic reduced representations, a type of vector symbolic architecture. DSHM bridges computational and neurally inspired systems by representing concepts as discrete units with compositional structure, while representing the relationships between concepts in a distributed manner.

Eliasmith's (2013) neural engineering framework provides a means of translating the linear algebra operations that define vector symbolic architectures into spiking neuron models. We do not implement DSHM in simulated neurons because it is computationally intensive and unnecessary for the work outlined in this paper. However, DSHM, or a close approximation, could be implemented as a neural model using the neural engineering framework, as discussed in Kelly, Mewhort and West (2014).

DSHM is based on Jones and Mewhort's (2007) BEAGLE (Bound Encoding of the Aggregate Language Environment), a model of statistical language acquisition rather than memory in general. Although memory is a component of both DSHM and BEAGLE, memory models and language models typically address different questions. For example, with BEAGLE, language learning requires exposure to an encyclopedic corpus over simulated years of the

agent's life. In contrast, psychology experiments on memory operate on small, simple sets of stimuli (e.g., word pairs) across relatively short amounts of time (minutes or days at the most). Another difference is that language requires complex processing (that linguists are still trying to work out) whereas psychology experiments generally hypothesize simple mechanisms that underlie memory, and test these mechanisms by observing simple behaviors such as recall or recognition.

What we will show in this paper is that the mechanisms used in BEAGLE can be repurposed to model declarative memory as it is studied in experimental psychology, indicating the potential for a single system that can integrate both types of research. We summarize the results of four sets of experiments that demonstrate the effectiveness of DSHM as a memory-modeling tool. Additionally, the choice of experiments illustrates the relationship DSHM has to connectionist architectures and symbolic architectures such as ACT-R. The model of the fan effect demonstrates the account of interference in DSHM. The model of the problem-size effect demonstrates the ability of DSHM to account for the frequency effect. The model of Paper, Rock, Scissors (PRS) play provides evidence that DSHM can be used to model tasks where the truth of facts change quickly and dynamically. The model of delayed feedback learning shows that DSHM is capable of learning a difficult binary decision-making task with probabilistic rewards, and thus is a competent model of utility learning.

BEAGLE

The following is a summary of the mechanics of BEAGLE, presented so as to be clear what features of DSHM are inherited from BEAGLE and which are new. For a less compact and more complete discussion of BEAGLE see Jones and Mewhort (2007).

Information Representation

BEAGLE is a holographic memory system, related to TODAM (Murdock, 1982; 1993). BEAGLE is a model of the mental lexicon. Given a corpus, BEAGLE develops a semantic representation for each word in the corpus. Each word is represented as a pair of vectors. Each vector consists of 2048 real valued elements (or dimensions) and has a Euclidean length of 1.0. One vector, called the environmental vector is a static internal representation of the word (an alphanumeric string being the external representation). The other vector, the memory vector, is dynamic in the sense that each time the system is exposed to that word in a new context (sentence), the word's memory vector is adjusted. The memory vector represents an accumulated history of all of the sentences in which the word has occurred. One can think of the environmental vector as storing one's knowledge of the orthography of the word while the memory vector stores one's knowledge of the sense (or meaning) of a word.

For the sake of simplicity, the environmental vector can be generated using random numbers sampled from a zero-mean normal distribution (as in Jones & Mewhort, 2007). Alternatively, environmental vectors can be constructed to reflect the details of a word's orthography (as in Cox, Kachergis, Recchia, & Jones, 2011; see also Hanngan, Dupoux, & Christophe 2011; Kelly, Blostein, & Mewhort, 2013).

Creating Associations When a sentence is 'read' by BEAGLE, a representation for each word in the sentences is created, if none exists already, by generating a random environmental vector and, by default, also initializing the memory vector to this same vector. Then, for each word in the sentence, associations between the word and the other words in the sentences are created.

These associations are themselves vectors, which are then added to the memory vector of the given word.

Encoding Co-occurrence BEAGLE is sensitive to two kinds of relationships between words. The first is simple co-occurrence. To encode that words have co-occurred in a sentence, the environmental vector of each word is added to the memory vector of each other word.

For example, if BEAGLE reads the sentence "cats run fast", the environmental vectors of 'run' and 'fast' are added to the memory vector of 'cats' (see calculation 1, where the '=' symbol is an assignment operator). The notation $\mathbf{e}_A$ is used to denote the environmental vector of A; $\mathbf{m}_A$ is used to denote the memory vector of A.

(1) $\mathbf{m}_{cats} = \mathbf{m}_{cats} + \mathbf{e}_{run} + \mathbf{e}_{fast}$

The memory vectors of run and fast are also updated (see calculations 2 and 3).

(2) $\mathbf{m}_{run} = \mathbf{m}_{run} + \mathbf{e}_{cats} + \mathbf{e}_{fast}$

(3) $\mathbf{m}_{fast} = \mathbf{m}_{fast} + \mathbf{e}_{cats} + \mathbf{e}_{run}$

As a result, the memory vector of each word will be similar to the environmental vectors of the other words. Similarity between vectors is measured by the vector cosine, which ranges from 1.0 for identical vectors, 0.0 for orthogonal vectors, to -1.0 for vectors pointing in the exact opposite direction. The cosine of one word's environmental vector and another word's memory vector will be statistically greater than 0.0 if they have co-occurred together, and will be approximately equal to 0.0 if they have not.

However, this type of encoding does not preserve any information about the relative order of the words.

Encoding Word Order When a sentence is imported into BEAGLE, the memory vector of each word is modified with a vector representing the positions of the other words in the sentence, relative to the given word. This is done as follows.

For each word in a sentence, which for a given instance will be referred to as the *target word*, a copy of the sentence is created with the target word replaced by a system-defined placeholder, **Φ**. This placeholder can be interpreted as meaning 'self'. That is, each word learns of the positions of other words relative to itself as 'self' and not using its own environmental vector representation.

For every subset of words surrounding (and including) the target word, within a maximum range of three words (on either side), the environmental vectors of the elements of the subset are combined via circular convolution (shown in Figure 1; Plate, 1995), and the result is added to the memory vector of the target word. The details of this process will be described using an example.

$$t_j = \sum_{k=0}^{n-1} c_k \, x_{j-k}$$

Figure 1. The function defining the circular convolution, **t**, of two vectors, **c** and **x**.

The following example describes the process of importing the sentence 'adventurous birds circle downwards' into BEAGLE. To make the example easier to read, the four words will be referred to by their first letters, in upper case.

When the sentence "A B C D" is imported, the memory vector of B (to pick an arbitrary example item) is modified as follows. First, B is made sensitive to the fact that it appears with

the three other words (i.e., co-occurrence). The environmental vectors of A, C, and D are simply added to the memory vector of B (as in the previous example).

Second, the positions of the other three words relative to B is encoded and added to the memory vector of B. The first step in doing this is to create a copy of the sentence from the perspective of the target word (B). This is done by replacing B with the placeholder vector **Φ**, resulting in the sentence "A Φ C D". Then, each subset of contiguous words that include Φ (i.e., 'A Φ', 'A Φ C', 'A Φ C D', 'Φ C', 'Φ C D') is encoded and added to the memory vector of B.

To encode the sequential order of items, an iterative binding process works left to right. A vector representing the left component is encoded as 'left' (or 'before') and a vector representing the right is encoded as 'right' (or 'after'). Encoding a vector as left (or, right) is accomplished by permuting the elements of the vector uniquely according to a predefined pattern. This makes the permuted pattern unique and distinct from the (non-permuted) environmental vectors of other items. The notation $l(\mathbf{v})$ is used to denote the encoding of a vector **v** as left; and $r(\mathbf{v})$ is used to denote encoding **v** as right. Once the left and right components are encoded as such, they are bound together via circular convolution. The symbol Θ is used in this paper to denote the circular convolution operator.

The encoding process starts with a left component consisting of the environmental vector of the first word in the set and the right component starts as the environmental vector of the second word. Thus, given the sequence 'A Φ C', the process would start by computing calculation 4.

(4) $l(\mathbf{e}_A) \Theta r(\mathbf{\Phi})$

On the next iteration, the left component is the result of calculation 4 and the right element is the third element of the set, as shown in calculation 5.

(5) $l(l(\mathbf{e}_A) \Theta r(\mathbf{\Phi})) \Theta r(\mathbf{e}_C)$

In general, the left component is the accumulated binding of the first *j* elements in the set and the right is the j+1th element. This binding process repeats until the last element has been bound to the rest. The vector representing the final result of binding a set of words together is added to the memory vector of the target word. The total change to the memory vector of B is presented in calculation 6.

(6) $\mathbf{m}_B = \mathbf{m}_B + \mathbf{e}_A + \mathbf{e}_C + \mathbf{e}_D$
$+ l(\mathbf{e}_A) \Theta r(\mathbf{\Phi})$
$+ l(l(\mathbf{e}_A) \Theta r(\mathbf{\Phi})) \Theta r(\mathbf{e}_C)$
$+ l(l(l(\mathbf{e}_A) \Theta r(\mathbf{\Phi})) \Theta r(\mathbf{e}_C)) \Theta r(\mathbf{e}_D)$
$+ l(\mathbf{\Phi}) \Theta r(\mathbf{e}_C)$
$+ l(l(\mathbf{\Phi}) \Theta r(\mathbf{e}_C)) \Theta r(\mathbf{e}_D)$.

The process would by applied for each of the words in the sentence read by BEAGLE.

Retrieving Associations The encoding algorithms described above are reversible, in that from the memory vectors of the words represented in the system, information about which other words a given word has occurred with can be extracted. When decoded, memory vectors are normalized to a Euclidean length of 1.0, prior to any operations performed upon them.

Retrieving Co-occurrence Frequency of co-occurrence is measured by the cosine similarity of the memory vector of a word to the environmental vectors of the other words in the lexicon. A high cosine indicates that the two words have co-occurred frequently. Two words that have never co-occurred will have a cosine near 0.0.

Retrieving Order That a word B has occurred immediately to the right of the word A can be determined by generating a probe vector equal to the second line of equation 6: $l(\mathbf{e}_A) \Theta r(\mathbf{\Phi})$.

This probe vector will have a cosine significantly greater than 0.0 with the memory vector of any word that has occurred to the right of A. This form of probing is referred to as *Resonance* (Jones & Mewhort, 2007).

Information can also be retrieved using the process of *Decoding* (Jones & Mewhort, 2007). Decoding uses *circular correlation*, the approximate inverse of circular convolution (see Figure 2). That a word A has occurred immediately to the left of the word B can be extracted from B by generating a probe vector from the memory vector of B that will have a high cosine to the environmental vector of any other word that has appeared to the left of B with sufficient frequency (see calculation 7). This probe is generated by applying the 'right' permutation (r) to the placeholder vector, $\mathbf{\Phi}$, and then correlating this vector with the memory vector of B. The inverse of the left permutation is applied to the result of the correlation (shown in Figure 2), resulting in a probe vector that will have a positive cosine with any environmental vectors of words that have appeared to the left of the word B. The symbol $\otimes$ is used in this paper to denote the circular correlation operator. The notation $l^{-1}(\mathbf{v})$ is used to denote applying the inverse of left permutation to vector $\mathbf{v}$ and $r^{-1}(\mathrm{v})$ is used to denote applying the inverse of the right permutation.

$$(7)\quad \mathrm{p} = l^{-1}(r(\mathbf{\Phi}) \otimes \mathbf{m}_{\mathrm{B}})$$

The resulting probe, p, is approximately equal to the environmental vector of A. The expression, 'approximately equal' is used for two reasons. First, correlation is an *approximate* inverse of convolution, i.e., it does not result in a perfect restoration of the original components. Second, the memory vector of a word can potentially encode thousands of associations, which adds noise to any decoding of a specific piece of information. Nevertheless, despite the noise,

decoding via correlation consistently preserves the appropriate relative ranking of matches via comparison of cosines.

$$y_j = \sum_{k=0}^{n-1} c_k \, t_{k+j}$$

Figure 2. The function defining the circular correlation, *y*, of two vectors, *c* and *t*.

Memory Vector Comparison The primary means of extracting information from the memory vectors of words are *decoding* and *resonance*, as detailed in the previous section. Using *decoding*, a memory vector is decoded to generate a probe that is compared to the environmental vectors of other words. Using *resonance*, environmental vectors are combined to form a probe that is compared to the memory vectors of other words.

A third method is to compare the memory vectors of words to each other. In BEAGLE, the memory vectors of words can be understood as points on a hyper-sphere. The synonymy of two words is inversely proportional to the distance between their memory vectors, where distance is measured by the cosine.

DSHM (Dynamically Structured Holographic Memory System)

DSHM (Dynamically Structured Holographic Memory System) is based on BEAGLE, but is modified to make it a general-purpose memory modeling system. Like BEAGLE, DSHM represents items by a pair of vectors: environmental and memory. However, unlike in BEAGLE, in DSHM the items need not represent words. Additionally, each item in DSHM has two additional variables, 'items' and 'ordered', which, respectively, store a list of sub-items and a flag

indicating whether that list is ordered or not. A DSHM item is considered to be an *atomic* item if it has no sub-items. An item is considered *complex* if it has sub-items.

Items

DSHM items, both atomic and complex, can be represented as strings. DSHM reserves the left and right bracket, the colon, and space symbols. Any other string of symbols is interpreted as a label for a DSHM item. Spaces delimit items that bear an unordered relationship to one another, and colons delimit items that bear an ordered relationship to one another. For example, “A B C” is interpreted as an unordered complex item with three atomic sub-items labelled 'A', 'B', and 'C'. The string “A:B C”, is interpreted as an unordered complex item with two sub-items, the first of which is an ordered complex item with two atomic sub-items A and B, the second of which is an atomic item C. The string “A:[B C]”, is interpreted as an ordered complex item with two sub-items, the first of which is A, the second of which is the unordered complex with two sub-items, B and C.

Defining an Instance of DSHM

DSHM has three global parameters: the dimensionality n; λ_o, which affects ordered item encoding; and λ_u, which affects unordered item encoding.

The dimensionality n is the number of elements in each environmental and memory vector. BEAGLE uses an n of 2048. This was found to be adequate for storing language information derived from a corpus approximately equal to the amount of text an average undergraduate student may have read in his or her lifetime (Jones & Mewhort, 2007). DSHM uses n as small as 128 to model some memory tasks described in this paper and elsewhere (Rutledge-Taylor, Pyke, West & Lang, 2010). Smaller n makes the storage of associations in the

memory vectors less reliable, which causes DSHM to make more mistakes and behave in a more stochastic manner.

The λ_o parameter determines the maximum distance two items can be from one another in the sub-item list of an ordered complex item and still be associated together. BEAGLE made use of a λ_o value of 3.

The λ_u parameter in DSHM is used when associating the sub-items of unordered complex items together. It determines the maximum size of subsets of sub-items that are bound together. There is no equivalent parameter in BEAGLE. However, BEAGLE behaves as if it uses a λ_u parameter value of 1.

Encoding Associations

Ordered Complex Items Ordered complex items are encoded in DSHM in a manner almost identical to how sentences are encoded in BEAGLE. However, unlike BEAGLE, DSHM does not use simple co-occurrence when processing an ordered complex item (i.e., DSHM does not add the environmental vector of each item to the memory vector of each other item). DSHM uses simple co-occurrence only for unordered items.However, the modeller can encode a sequence as both order and unordered if he or she has a theoretical motivation to do so (e.g., to replicate BEAGLE lexicon acquisition).

For example, when the ordered complex item "[A:B:C:D]" is imported into a DSHM model with an λ_o of 2, the memory vector of A would be updated according to calculation 8.

$$(8)\ \mathbf{m}_A = \mathbf{m}_A + l(\mathbf{\Phi}) \Theta r(\mathbf{e}_B) + l(l(\mathbf{\Phi}) \Theta r(\mathbf{e}_B)) \Theta r(\mathbf{e}_C)$$

The memory vectors of the other words would be updated in a similar manner. Adding convolved sets of environmental vectors to a memory vector makes the updated memory vector

sensitive to the co-occurrence of particular combinations of other items, rather than just those other items individually.

Unordered Complex Items Unordered complex items are encoded as follows. First, a list is created composed of every subset of the unordered complex item with λ_u+1 or fewer sub-items. For each item in each subset in the list, the environmental vectors of the other items in the subset are convolved together, and the result is added to the memory vector of the item.

For example, if the complex item "[A B C D]" is imported into a DSHM model with a λ_u of 2, the list of subsets of items to be associated together would be: [A B C], [A B D], [A C D], [B C D], [A B], [A C], [A D], [B C], [B D], [C D]. For the subset [A B C], the convolution of the environmental vectors of B and C would be added to the memory vector of A; the convolution of the environmental vectors of A and C would be added to the memory vector of B; and the convolution of the environmental vectors of A and B would be added to the memory vector of C.

The update to the memory vector of A for the complex item "[A B C D]" is presented in calculation 9.

$$(9)\ \mathbf{m}_A = \mathbf{m}_A + \mathbf{e}_B + \mathbf{e}_C + \mathbf{e}_D + \mathbf{e}_B \Theta \mathbf{e}_C + \mathbf{e}_B \Theta \mathbf{e}_D + \mathbf{e}_C \Theta \mathbf{e}_D$$

Nested Complex Items Complex items that contain nested complex items are encoded recursively. Nested complex items are encoded as either ordered or unordered complex items, as previously described. Then, to encapsulate the information within the nested complex items, they are encoded as complex. Encoding a vector as representing a complex item is accomplished by permuting the elements of the vector according to a predefined random pattern (like how vectors are encoded as left or right). The resulting vectors are then incorporated into the complex item that contains them in the same way as atomic items. The notation $c(\mathbf{v})$ is used to denote the permutation of a vector $\mathbf{v}$ to indicate it is complex; c^{-1} is the inverse permutation. For example, if

the complex item "[A B [C:D]" is imported into DSHM with $\lambda_u = 2$ and $\lambda_o = 2$, then the memory vector of A is updated as follows:

$$\mathbf{m}_A = \mathbf{m}_A + \mathbf{e}_B + c(l(\mathbf{e}_C) \Theta r(\mathbf{e}_D)) + \mathbf{e}_B \Theta c(l(\mathbf{e}_C) \Theta r(\mathbf{e}_D)) \quad (10)$$

Permuting the vectors representing complex sub-items allows DSHM to distinguish [A [B C] D] from [A B C D].

Cardinal Items As a general model of memory, DSHM needs to account for a variety of mental representations. As such, we have found it convenient to create universal atomic items, called *cardinal items* that represent special concepts. For example, the model of the problem-size effect uses cardinal items representing *true* and *false*, and the model of delayed feedback learning uses cardinal items representing *good* and *bad*. Computationally, these items behave like any other. They simply have a special status in the model as representing concepts internal to the model, rather than external stimuli.

Decoding Associations

The memory vectors of items in DSHM can be decoded to determine which items they co-occurred with. This is done via pattern completion. Here it is necessary to define some terms.

Query Item A query item is an unspecified item that acts like a variable. It is represented by a string preceded with a question mark, e.g., '?x'. A query item is an atomic incomplete item.

Complete Items and Incomplete Items Previously we have distinguished between two types of items: atomic and complex. Here we make an additional distinction: complete items and incomplete items. A complete item is an item with all of its sub-items fully specified. Atomic items (except query items) are considered complete items.

An incomplete item is defined recursively. A query item is an atomic incomplete item. A complex item with a query item as a sub-item is an incomplete item (e.g., [A ?x]). A complex item with a sub-item that is an incomplete item is an incomplete item (e.g., [A [B ?x]]).

Context Item The complete sub-items of an incomplete item are referred to as context items because they provide the context for determining how best to complete the incomplete item.

Completing an Incomplete Item Associations are decoded by presenting an incomplete item to a DSHM model for completion. The sources of information for completing the item are the context items (i.e., the complete sub-items) of the incomplete item. The result of item completion is a rank ordered list of candidate completions.

Recursive Completion The completion process is recursive. The base case is when the item to complete is a query item. Initially, every item in the system is an equally good candidate, thus, the list of completions of the query item is every item in the system.

When the item to complete, called the *root* item, has a mix of complete and incomplete sub-items, the context items are decoded to produce a set of probe vectors that represent the knowledge residing in each of the complete items about what other items, in the positions of the incomplete items, have co-occurred with them.

These probe vectors are used to evaluate the degree of association between each of the candidate completions of the incomplete sub-items and the context items. A new list of completions of the root item is compiled based on which items best match the probes.

Generating Probes Probe vectors are generated in a manner analogous to how they are produced in BEAGLE. As is the case in BEAGLE, all of the encoding operations in DSHM are reversible.

Given an unordered incomplete item [A B ?x], four probe vectors would be produced (provided the λ_u parameter is equal to 2 or more), according to calculations 11 to 17:

(11) pAe = $\mathbf{m}_A$
(12) pBe = $\mathbf{m}_B$
(13) pABe = $\mathbf{e}_B \otimes \mathbf{m}_A$
(14) pBAe = $\mathbf{e}_A \otimes \mathbf{m}_B$
(15) pAm = $\mathbf{e}_A$
(16) pBm = $\mathbf{e}_B$
(17) pABm = $\mathbf{e}_A \oslash \mathbf{e}_B$

The label for a probe (in this paper) follows the convention of starting with a lower-case 'p', and ending with a lower-case 'e' if it is designed to match the environmental vector of potential completions, and a lower-case 'm' is designed to match the memory vector of potential completions. The first of the infix capital letters indicates which context item's memory vector is the source of information (i.e., which memory vector is being decoded). Subsequent letters indicate other context items used in decoding the memory vector of the first context item.

$$\text{Cosine} = 1 / (x + 1)^{1/2}$$

Figure 3. The formula for calculating the expected cosine correlation between the memory vector of a word A, and the environmental vector of a word B, where x is the number of vectors that have been added to $\mathbf{m}_A$ since $\mathbf{e}_B$ was added to $\mathbf{m}_A$.

Given an ordered incomplete item [A:B:?x], five probe vectors would be produced (provided the λ_o parameter is equal to 2 or more), according to calculations 18 to 22:

(18) $pABe = r^{-1}(l(\mathbf{e}_B) \otimes (l(\mathbf{\Phi}) \otimes \mathbf{m}_A))$
(19) $pBe = r^{-1}(l(\mathbf{\Phi}) \otimes \mathbf{m}_B)$
(20) $pBAe = r^{-1}(l(\mathbf{\Phi}) \otimes r^{-1}(l(\mathbf{e}_A) \otimes \mathbf{m}_B))$
(21) $pAm = l(l(\mathbf{e}_A) \Theta r(\mathbf{e}_B)) * r(\mathbf{\Phi}))$
(22) $pBm = l(\mathbf{e}_B) \Theta r(\mathbf{\Phi})$

Once all probe vectors have been computed, all the suffix 'e' probes are added together and used as an aggregate environmental vector probe to perform *decoding*. Similarly, all the suffix 'm' probes are added together (and then normalized) and used as an aggregate memory vector probe to perform *resonance*. The result of either decoding or resonance is a ranked list of candidate completions. The items are ranked by the cosine of the aggregate probe with either the environmental or memory vectors of the items, depending on whether *decoding* or *resonance* is used.

Recursive Completion Example Hierarchically structured incomplete items are completed in a depth-first manner, with completions propagating up from the atomic items to the root of the incomplete item. For example, to complete [?x [G:?y]], first completions of [G:?y] are generated. To do this, probes are created based on G: $pGe = r^{-1}(\mathbf{m}_G \otimes l(\mathbf{\Phi}))$ and $pGm = l(\mathbf{e}_G) \Theta r(\mathbf{\Phi})$. Either or both of these can be used to find the best substitutions for ?y. If for example, the top ranked completions of [G:?y] are {[G:H], [G:J]}, this would indicate that there were only two strong candidate completions of [G:?y] and that [G:H] is the stronger of the two (in virtue of occurring first in the list). These sub-item completions would each, separately, be used to generate possible competitions of [?x [G:?y]]. For example, probe to test for ?x given [G:H],

would be: pGHm = $c(l(\mathbf{e}_G) \Theta r(\mathbf{e}_H))$. Note that because complex sub-items do not have memory vectors, decoding is not an option for generating probes for ?x. The final completions are scored by combining the scores of the sub-item completions and the scores of the root item completion. The details of this score combination function will not be given here. However, the basic consequences are that the final list of completions will be ranked according to this combined score. For example, the final list of completions of [?x [G:?y]] might be: {[A [G:J]],[B [G:H]]}. This would suggest that although [G:H] is a stronger completion of [G:?y] than is [G:J], [A [G:J]] is a stronger final completion because [G:J] is more strongly associated with A than [G:H] is with B.

Models

In this paper we present four DSHM models of various cognitive phenomena. In each case, previously collected empirical results are used as the basis for developing the DSHM models. Comparisons to models created using other systems are made where such models are available.

The Fan Effect

The fan effect is a well-known memory phenomenon relating to the time required to affirm (or, reject) a given proposition as true. The effect was first described by Anderson (1974). Anderson generated a study set of 26 sentences each of which described a fact about one of 16 persons residing in one of 16 places (e.g., “the hippie is in the park”). The fan of a word is the number of sentences from the study set in which it appears. The fan of each person and place in the study set is one, two or three. Anderson had each experimental participant memorize the study set, and then measured the time that elapsed between the subsequent presentation of a sentence and the participant's signal that the sentence was either a member, or not a member of

the study set. The results showed that there was a positive correlation between reaction time and the fan of the sentence (the sum of the fans of each of the content words in a test sentence).

As a basic phenomenon of human memory, the fan effect has served as a useful paradigm for testing theories of memory retrieval, including those pertaining to recall time and accuracy. DSHM is primarily a theory of how information is represented and structured in memory. However, as such, it also offers mechanisms for retrieving information from memory. Therefore, it ought to be consistent with the prevailing theories of memory retrieval. Mechanisms such as spreading activation (Anderson & Reder, 1999), processes such as retrieval effects (Anderson & Reder, 1999), representation effects (Goetz & Walters, 2000; Radvansky, et al., 1993), and memory capacity (Bunting et al., 2004) have all been proposed as responsible for, or playing a role in, producing the fan effect. The basic mechanisms in DSHM are cue (i.e., probe) based retrieval and interference. However, informational structures extracted from DSHM can be interpreted as having a particular activation value and are subject to spreading activation

Memory Retrieval

Traditionally, memory retrieval can be divided into two types: recall and recognition. In general terms, recall can be understood as retrieving some content from memory that matches some cue, such as the content provided in a question like “where are the hippies?”. Here, the words ‘where’ and ‘hippie’ cue the recall of places known to be populated by hippies. Recognition can be understood as affirming that a presented fact or item is known, true, or has been studied. Asking a participant whether he or she has studied the proposition ”the hippie is in the park” is an example of a recognition memory task. In this case, the proposition as a whole and each content word in the proposition is a potential cue to memory. Participants of experiment

1 of Anderson (1974) were tasked with both recall and recognition: recall during the *learning* phase and recognition during the *reaction-time* phase.

A model of memory must explain the mechanisms that underlie both recall and recognition. These mechanisms must account for how these processes can both succeed and fail. In the case of recall, success is a matter of only retrieving correct answers to questions, or otherwise providing only the appropriate matches to the provided cues. Failure is to either omit correct responses, or retrieve incorrect responses. Successful recognition is a matter of affirming true facts and correctly rejecting false ones, while failure is to deny true facts or to affirm false ones.

Where are the Hippies?

During the learning phase of Anderson's fan effect experiment, participants were first asked to rehearse the study set of sentences. To confirm that they could remember each of the sentences, the participants were asked questions of the form "who is in the park?" and "where are the hippies?" (Anderson, 1974). For each question there is exactly one, two or three correct answers, depending on the fan of the content word supplied in the question. A challenge to any model of memory is to explain how all and only the correct answers to a question are recalled.

To ask the DSHM system to answer a recall question an incomplete pattern encoding the question is presented to it. For example, to ask the system "where are the hippies?", the incomplete pattern [hippie:?x] is presented to it. As discussed above, every item in the system is assigned a value, corresponding to how well it matches the cue (in this case 'hippie'). As a matter of how the state-space of the system organizes itself, only items that have co-occurred with the cue will be activated to a significant extent. Thus there is a distinctive drop-off in activation values between those items associated with the probe and those that are not. An

important assumption that we have made is that the items before the drop are recalled and the items after the drop are not.

Table 1 displays the top six answers to the questions: “where are the hippies?”, and “where are the lawyers?”, presented to a DSHM system with 2048 dimensional vectors, trained only on the study set. The word ‘hippie’ has a fan of one, while ‘lawyer’ has a fan of three. Therefore, there is only one correct answer to former question, and three correct answers to the latter. A *Cut-off* mechanism in DSHM automatically prunes the responses after the first significant drop-off in score when traversing the list, sorted in descending order of score. The system correctly scores only one answer to the first question highly, and three to the second. This precise response profile corresponds to the behavior of an experimental participant with extremely good memory. Errors can be produced by a model that uses fewer dimensions, corresponding to a participant with poorer memory. Using vectors with fewer dimensions limits the memory capacity of the model, and makes it more vulnerable errors caused by memory interference.

Table 1: Where are the hippies?

Answers to questions posed to DSHM			
Where are the hippies?		Where are the lawyers?	
Answer	Value	Answer	Value
[hippie:park]	0.699	[lawyer:store]	0.501
[hippie:store]	0.007	[lawyer:park]	0.487
[hippie:captain]	0.003	[lawyer:bank]	0.475
[hippie:cave]	0.001	[lawyer:fireman]	0.010
[hippie:debutante]	0.001	[lawyer:cave]	0.003
[hippie:church]	0.001	[lawyer:captain]	0.002

The drop-off in activation values serves as a mechanism for determining which items the system will recall, and which it will not. This makes intuitive sense. When a human participant is asked “where are the lawyers?”, he or she will not answer with a rank ordered list of all 16

places mentioned in the study set. Rather, he or she will respond with only a few answers that seem more correct than the other possible answers. The ability of DSHM to provide several appropriate answers to a single question, when more than one answer is correct, is one of the strengths of the system.

Is the Hippie in the Park?

During the test phase of the experiment, a set of test sentences is presented to the participant, one sentence at a time. The task of the participant is to decide as quickly as possible, while maintaining a high degree of accuracy, whether the presented sentence was a member of the study set or not (Anderson, 1974). The DSHM model makes this membership judgment not by searching memory for the presented proposition, rather it assesses how tightly associated the content words in the proposition are. This is done by asking itself "where are the hippies?", and "who is in the park?". The average of the activation values associated with the correct answers (i.e., 'park', and 'hippie') determines the activation of the proposition as a whole. An incorrect rejection of a study set sentence occurs when the model fails to recall due to the cut-off mechanism described above.

Spreading Activation

Pattern completion can be interpreted as a form of spreading activation in which activation spreads out from the probe and weakens as it spans the state-space (i.e., the closer to the probe, the higher the activation). Thus the match value would correspond to a cue specific activation level. We interpret activation level as determining how much time it takes for each item to be retrieved (in a manner similar to ACT-R).

In the case of foils, the model does not know in advance how many items will be recalled by the 'who' and 'where' sub-questions. Therefore, there may be a hesitation associated with

waiting to make sure that no more items (possibly including a match) are forthcoming. For example, if presented with the foil "the hippie is in the store", the sub-question "where are the hippies?" will quickly cause the recall of 'park'. However, the fact that 'store' will not be recalled cannot be verified until all of the activated items have been recalled. Thus, a brief period when no more items are recalled must follow the last recalled item to ensure that the cut-off has been reached. Here, an analogy to cooking microwave popcorn can be made. When cooking microwave popcorn, the instructions typically state that the popcorn bag is to remain in the oven while the gaps between pops do not exceed a few seconds. The need to keep the bag in the oven can be verified the instant a pop is heard. However, in order to decide not to keep the bag in the microwave any longer, one needs to wait a few seconds after hearing the latest pop to verify that the required time between pops has elapsed. By analogy, to reject a foil also requires a hesitation to confirm that the target proposition is not about to 'pop' into mind.

DSHM Model of the Fan Effect

The DSHM model of the fan effect follows the original conditions specified in experiment 1 of Anderson (1974). The sentences were represented as complex items composed of an ordered list of two items representing a person and place, e.g., [hippie:park]. The omission of function (aka stop-words) words in the representation of the sentence (as is common in text analytics) does not significantly affect the important features of the memories for these sentences; they are redundant to the information contained in the content words, in this case.

Special care was taken to ensure that the simulated participants followed as closely as possible the exact procedures followed by the human experimental participants. Both human participants and the DSHM models perform the experiment in three phases:

1) *Study phase*: rehearse the study set;

2) *Learning phase:* confirm knowledge of the study set by answering recall questions of the form "where are the hippies?";
3) *Reaction-time phase:* answered recognition questions of the form, "was 'the hippie is in the park' a member of the study set?".

During the learning phase, an error is made when a participant (either human or simulated) offers too few correct answers (e.g., answering "the lawyer is in the store", and "the lawyer is in the park", but omitting "the lawyer is in the bank"), or at least one incorrect answer (e.g., responding with the correct answer, "the hippie is in the park", but also with the incorrect answer, "the hippie is in the store"). When at least one error is made, a participant is made to rehearse the propositions that were recalled incorrectly and then retested.

A parameter we chose to manipulate is how much background knowledge each simulated participant has preloaded before learning the study set. Background knowledge was represented as a variable amount of [person:*random number*], and [*random number*:place] items (e.g., [hippie:7845]). The random numbers represent arbitrary facts the participant may know about the places and persons, related to those in the study set, in advance of the experiment. For example, the participant may know that hippies live in an apartment in his or her building, or that there are pigeons in the park, etcetera. We hypothesized that background knowledge could affect a model's ability to distinguish between sentences from the study set and foils by increasing the number of facts that may interfere with one another in the system. It also adds an element of ecological validity to the simulations in that no human participant enters into the experimental scenario without some knowledge relating to the persons and places appearing in the study set.

Generating Reaction Times

Reaction times are derived from the activation values of the sentences produced by the system. At present we have not developed a universal function that relates activation to reaction time. Here we use ad hoc formulae to make comparisons between the DSHM models and the human data from Anderson (1974). For affirmations we used the formula: 350ms + 550ms * 1/*a*; where *a* is the activation of the affirmed proposition. For rejections we used the formula: 450ms + 550ms * 1/*r*; where *r* is the activation of the rejected proposition. The additional 100 milliseconds added to the reaction times for rejections is intended to compensate for the characteristic delay associated with judging foils, as previously discussed.

Generating Errors

Human participants incorrectly judged the study set membership of a test sentence 3.9% of the time in experiment 1 of Anderson (1974). Thus, an adequate model of the fan effect should make a similar number of errors. Since errors in DSHM systems only occur when the memory capacity of the system is saturated, a range of vector dimensionalities were tested to determine which makes errors with a frequency approximating that of human participants.

The Simulations

The two parameters discussed above were manipulated so as to produce several populations of models, each with common parameters. Vectors with {32, 128, 512, 2048} dimensions and background knowledge parameters in the set {0, 1, 2} were used. The background knowledge parameter is equal to the number to additional random propositions the model is trained on for each of the 16 persons and 16 places; e.g., for a value of 1, one random fact of the form [person:*random number*], for every person, and similarly for every place, was

entered into the model's memory prior to learning the study set. In the case of no background knowledge, the study set was initially read once during the study phase, and twice otherwise. This ensured that every sentence in the study set was read at least one time more than any of the background knowledge sentences.

The combinations of these parameters make 12 unique sets of simulations. There were 100 simulated participants for each set.

Results

For each of the 12 sets of simulations, we recorded the mean reaction times and error rates for each of the nine kinds of sentences. The different kinds of sentence are differentiated by the fans of the content words. The person and place appearing in each sentence has a fan of one, two or three; this makes nine combinations. Data was collected for both true sentences, those appearing in the study set, and for false sentences, or foils, those not appearing in the study set.

There were several general patterns that emerged from these simulations: 1) mean reaction time was somewhat larger for simulations that used vectors with fewer dimensions; 2) the standard deviation of the reaction times was dramatically larger for simulations that used fewer dimensions; 3) judgment errors were similar to human performance for only simulations that used 128 dimensional vectors; 4) although the amount of background knowledge did not significantly affect mean reaction time, it did increase the standard deviation of the reaction times for all dimensionalities, except for simulations with 32 dimensional vectors; 5) the behavior profile of the models was approximately equal for both true sentences and foils.

Figure 4, illustrates points 1 and 2 above. In order to simplify the figure, only the true sentences with fans of two and six are depicted. The fan of a sentence is the sum of the fan of

the person and place. Figure 4 illustrates the fan effect pattern in the data, for each dimensionality. The bars indicate the standard deviations on these mean reaction times in milliseconds. The figure truncates the standard deviation of 3671 milliseconds for the simulations with 32 dimensional vectors and fan of six. This decrease in the fan effect with dimensionality is consistent with studies showing a decrease in the fan effect with working memory capacity (Bunting et al., 2004).

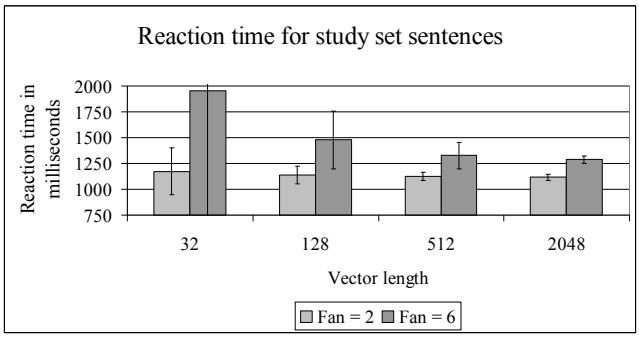

Figure 4. Reaction time for recalling study set sentences.

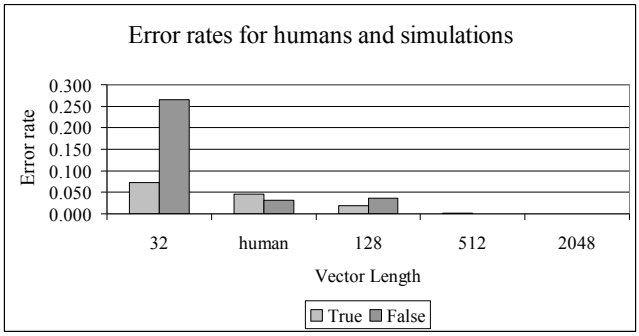

Figure 5. Error rates.

Figure 5, which includes human data from Anderson (1974), illustrates our third point: Models that used vectors with 128 dimensions were the only models that produced an error rate resembling human performance. Additionally, and informally, the standard deviation of the reaction times for the 128 dimensional models has a 'human-like' profile, whereas models with

other dimensionalities do not. Unfortunately, Anderson (1974) did not include standard deviations on his human data, and so we could not compare the human data and the model on this measure.

With respect to our fourth point, the effect of background knowledge, it should be noted that had the model not read the study set multiple times during the study phase of the experiment, the background knowledge would have affected the model's performance more dramatically. This is explained by the frequency effect discussed in the *Math Cognition* section of this paper.

Tables 2 and 3 present the mean reaction time and error rate data for the models with 128 dimensional vectors, in the same format as the human data presented in Anderson (1974). The rows are organized by the fan of the place, while the columns are organized by the fan of the person. Each cell contains a mean reaction time in milliseconds and an error rate for propositions with the fans corresponding to the given cell. As with Anderson (1974), only the times corresponding to the affirmation of true sentences and the rejection of false sentences were included in the reaction times; i.e., the times associated with errors (misses and false positives) were not included.

Table 2: Model of trues

Reaction times and error rates for trues

	Person fan			
Place fan	1	2	3	Mean
1	1137	1191	1240	1189
	000	002	000	001
2	1166	1285	1377	1276
	002	027	038	022
3	1232	1371	1478	1360
	000	032	072	034
Mean	1178	1282	1365	1275
	001	020	037	019

Table 3: Model of falses

Reaction times and error rates for falses

	Person fan			
Place fan	1	2	3	Mean
1	1170	1225	1320	1238
	003	020	042	022
2	1242	1288	1437	1322
	018	023	048	030
3	1323	1413	1488	1408
	045	057	072	058
Mean	1245	1308	1415	1323
	022	033	054	036

Evaluation of the Model

The reaction times for the DSHM model were generated according to ad hoc functions designed to produce values that match the human data, on average. Thus, gross agreement between the DSHM reaction times and the human reaction times is a given. However, the functions are linear and do not finesse the data so as to produce the fan and min effects (Anderson, 1974). Thus, a comparison between the DSHM model and the human data from Anderson (1974) is provided. Figures 6 and 7 illustrate the close match in the pattern of the reaction times, for sentences with fans of two, three, four, five, and six for true sentences and false sentences respectively. Also, it should be noted that the error bars represent predictions of what the level of variability in the human results should be, but this could not be confirmed as that data was not available.

Relationship to the ACT-R Model

Although relying on very different mechanisms, the DSHM model is consistent with the ACT-R model of the fan effect (Anderson & Reder, 1999). For both models, the characteristic pattern of reaction times defining the fan effect results from patterns in the activation levels of the propositions learned by the systems.

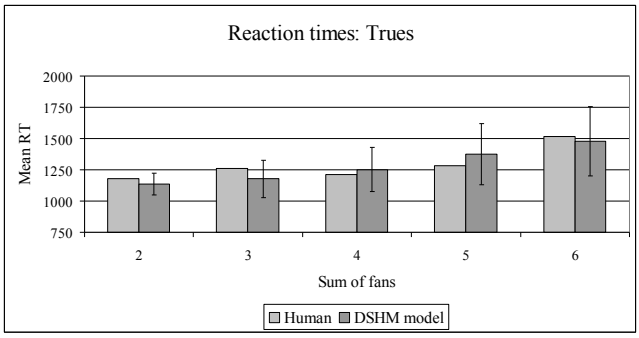

Figure 6. Reactions times for study set sentences. Bars indicate standard deviation.

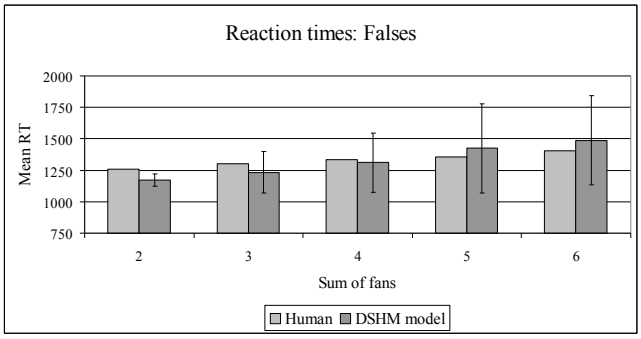

Figure 7. Reaction times for foils. Bars indicate standard deviation.

The activation of a target sentence in DSHM is a function of the association between the person and the place appearing in the sentence. Mathematically, this is a measure of the distance between the environmental vector of each word, properly encoded, and the memory vector of the other word in the state-space defined by the system.

The activation of a target sentence according to the ACT-R model is a function of the associations between the chunk representing the target sentence and the chunks representing each of the person and place. Mathematically, it is equal to the sum of a base level of activation and a weighted sum of the activations of the person and place. In the case of foils, the target sentence does not exist in the declarative memory system of the model and cannot be retrieved. Instead, a chunk, partially matching the target, which contains one of the two content words is retrieved. As a result of the retrieved chunk and the target sentence chunk having only one of the person

and place in common the net activation of the retrieved chunk has one less source of activation, and thus, on average, has a lower net activation value than the average true sentence.

Anderson and Reder (1999) do not discuss how errors could be produced by the ACT-R model. Presumably, errors could be caused by increasing the amount of noise in the system, and/or raising the activation threshold for recall.

We are of the belief that the DSHM system and the ACT-R architecture are compatible. The DSHM and ACT-R models of the fan effect both agree with the human data. This agreement is a result of DSHM and the declarative memory system of ACT-R providing alternative accounts of the same psychological memory mechanisms. We claim that ACT-R provides a higher-level, more abstract, account of memory, making use of mathematical formula to describe the activations of chunks in memory. DSHM, on the other hand, produces the same profile of behavior as an emergent property of the distributed representation of associations between items in the state-space of the system. For example, ACT-R activation can be interpreted as emergent on DSHM association strength, and ACT-R noise as emergent on DSHM vector dimensionality (memory capacity). As such DSHM resides hierarchically between ACT-R as an abstract computational model of memory and lower-level neural network models of memory, such as those described in Goetz and Walters (2000).

Math Cognition (and the Problem Size Effect)

In this section we used DSHM to model Zbrodoff's (1995) findings on the problem size effect, a well-known effect in the area of math cognition. The data showed the effects of manipulating both frequency and interference. We successfully modeled this using DHSM,

demonstrating that frequency and interference effects arise naturally as a function of how holographic systems work.

One function that DSHM models well is memory interference. As we have shown in the previous section, the *fan effect* (Anderson, 1974) falls naturally out of the DSHM architecture. The fan effect addresses only the effect of inter-fact 'interference' on the efficiency of fact retrieval. However, there is another factor that also strongly impacts retrieval speed/efficiency: the person's frequency of exposure to that fact. For example, if a participant reads "the lawyer is in the store" once and "the lawyer is in the bank" four times, the fans of 'store' and 'bank' are each still one. However, one would expect that the association between 'lawyer' and 'bank' to be stronger than the association between 'lawyer' and 'store'. Thus, both fan effects and frequency effects impact the efficiency of fact retrieval.

In ACT-R, frequency of exposure is represented separately by the base level activation function (Anderson & Lebiere, 1998). In DSHM frequency produces an effect by causing a fact to be pulled more in one direction than another. For example, if the lawyer was in the store more often than in the bank, the vector representations of the lawyer would be end up closer to that of the store than that of the bank. To test the interaction of frequency and fan in DSHM we modeled the data of Zbrodoff (1995), who manipulated both of these in the context of studying arithmetic cognition (i.e., retrieving simple facts such as "2 + 3 = 5").

Zbrodoff's Experiments

In arithmetic cognition research, it is often found that small sums, like 2 + 3 = 5, are more quickly retrieved than large sums, like 5 + 7 = 12. This is the so-called *problem-size effect* (reviewed by Zbrodoff & Logan, 2005). It is known that small problems are presented more

frequently in math texts than large problems (Hamann & Ashcraft, 1986), so this effect could be due to frequency of exposure, however, interference between memory elements has also been proposed as an explanation (Seigler, 1987; Verguts & Fias, 2005). Zbrodoff's (1995) goal was to investigate the extent to which different retrieval times for different math facts should be attributed to fan effects (which she termed 'interference') or frequency effects (which she termed 'strength') or an interaction of these two effects.

Zbrodoff (1995) conducted four experiments that manipulated the effects of strength and interference to assess their relative contribution to the problem-size effect. In each experiment participants were shown mathematical problems with a potential answer on a computer screen. The participant's task was to press one key if the problem was correct, and press a different key if the problem was false. To manipulate frequency and interference for facts in memory, and to eliminate pre-experimental practice effects, instead of using regular arithmetic, Zbrodoff's stimuli were alphabet arithmetic facts (e.g., A + 3 = D, which indicates that the number three letters past A is D). The first addend was always a letter of the alphabet; the operator was always addition; the second addend was 2, 3 or 4; and, the sum was a letter of the alphabet. The problem was considered true if translating the letters to numbers according to their index in the alphabet resulted in a true math fact. For example, "A + 2 = C" is true because translating 'A' to 1 and 'C' to 3, results in the true math fact "1 + 2 = 3". Participants were told that they could determine whether a problem was true or false by starting with the first addend (e.g., A) and then counting through the alphabet the number of letters specified by the second addend (e.g., 3).

In each experiment, participants were exposed to large blocks of problems, each of which consisted in repeated instances of a group of 12 unique problems. Each group consisted of six true and six false problems. A false problem was a problem for which the answer was incorrect.

In Experiments 1 and 2, each problem consisted of the combination of one of six letter addends with one of the digit addends (2, 3 or 4). Each letter was paired with only one digit, and each digit was paired with two letters. False problems were generated by setting the incorrect answer to one letter past the correct answer (e.g., B + 3 = F). The groups of problems were counterbalanced so that each letter addend was paired with each numerical addend equally frequently. Table 4 provides an example of a group of problems for Experiments 1 and 2.

Table 4: Experiment 1 & 2 group of problems

Letter addend		Number addend		True answer	False answer
A	+	2	=	C	D
B	+	3	=	E	F
C	+	4	=	G	H
D	+	2	=	F	G
E	+	3	=	H	I
F	+	4	=	J	K

Groups of problems for Experiments 3 and 4 were similar to those for Experiments 1 and 2 with two changes. In each stimulus set only two unique letter addends were used (e.g., A and B). Each letter was paired with each of the three digit addends. The false answers were either one letter past the correct answer or one letter before it. Table 5 provides an example of a group of problems for Experiments 3 and 4.

Table 5: Experiment 3 & 4 group of problems

Letter addend		Number addend		True answer	False answer
A	+	2	=	C	D
A	+	3	=	D	E
A	+	4	=	E	F
B	+	2	=	D	C
B	+	3	=	E	D
B	+	4	=	F	E

Re-Analysis of Zbrodoff's Experiments

Zbrodoff (1995) used a method of analyzing the data that was not useful for our purposes. Specifically, she plotted a straight line through the reaction times associated with the different addends and used the slope of this as an index of the magnitude of the Problem Size effect. This is a common way of analyzing the Problem Size effect within the area of Math Cognition. However, since we were interested in testing our model, not the Problem Size effect, we re-analyzed the reaction time data (which is provided in the paper) and came up with a somewhat different interpretation of the results.

Experiment 1

Experiment 1 was focused on learning in the short term. In it participants were presented with three blocks of problems. Each consisted of 96 true and 96 false problems. This was problematic for us because the learning process involved initially doing the calculations to get the answers. Therefore, these blocks represent a mixture of calculating and memorizing. By the third block we assume participants were using memory, but may have still been relying on calculation as well.

Since DSHM models memorize only, we could not represent the effect of calculating. Hence, our goal was not to model the learning curve. Instead we were focused on the long term learning trends. Because of this, we did not include Experiment 1 in our analysis. However, the results of Experiment 1 showed that frequency of exposure to a problem eventually resulted in faster reaction times, regardless of whether the addend was large or small. This is consistent with the DSHM model since, if all other things are held equal (as they were in Experiment 1), more exposures results in faster recall.

Experiment 2

The purpose of Experiment 2 was to determine whether the frequency effect demonstrated in Experiment 1 held up once performance had reached asymptote. That is, with enough practice does response time performance converge despite differences in frequency of exposure? To test this, participants were presented with 15 blocks of problems (3

blocks per day) identical to those from the *Standard* condition of Experiment 1, which was designed to mimic real world math learning conditions where smaller numbers are encountered more frequently than larger numbers. To accomplish this real world problem frequency, the addend 2 problems were presented 24 times per block, the addend 3 problems were presented 16 times per block, and the addend 4 problems were presented 8 times per block.

The results showed that performance did converge as it reached an asymptote. However, we also noticed that reaction times were much lower at asymptote in Experiment 2 than in Experiments 3 and 4 (approximately 600 msec in Experiment 2; 1000 msec in Experiments 3 and 4). To explain this discrepancy we examined the stimuli and found that in Experiment 2 each letter-answer was uniquely associated with a different letter-addend in the question. For example, the addend 'A' is associated only with the sum 'C' (see Table 4). Therefore participants could memorize a pairing between the letter-addend in the question and the letter-answer. Given that this was not the case in Experiments 3 and 4 we feel the learning and use of this strategy is a likely explanation for the faster reaction times in Experiment 2. As such, we chose not to model experiment 2.

Experiment 4

We created DSHM models of both experiments 3 and 4. However, since experiment 4 is simpler, we present our results of this model first. In Experiment 4 all of the problems in Table 5 were presented with equal frequency. To model this, each problem, including the answer and whether the answer was true or false, was encoded as an item and presented to a DSHM model.

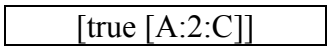

Figure 8. An example DSHM item representing an addition fact.

Each item was unordered and was composed of two sub-items: a cardinal item representing the truth of the problem, and a complex item representing the problem. The item representing the problem was ordered and consisted of atomic items representing the first addend, second addend, and the sum. Each item was presented to the DSHM model once for every exposure by the human participants to the corresponding problem.

There were two ways the model could decide if a question was true. One was to give the model a complete representation of the problem, and have it decide whether it was true or not. To do this, the model was presented with an incomplete item with a structure similar to that in Figure 8, but with the truth item replaced with a query item.

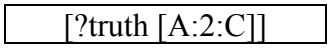

Figure 9. An example DSHM item used for recalling the truth of an addition fact.

The model would then return a completion with the query item replaced with either true or false. The second way was to submit the question with the answer replaced with a query item and the truth component set to true.

[true [A:2:?sum]]

Figure 10. An example DSHM item used for recalling the correct sum of an addition fact.

In this case the model would return what it believed to be the correct answer. The model can make errors but this data is not presented here. The second method fit the data better than the first, suggesting that when presented with a problem, the participants were determining what the correct sum should be and making a comparison of this value to the sum presented. In this case the model makes the same predictions for true and false questions. Consistent with this, the human reaction times for the true and false questions were very similar. To get accurate reaction times from the model, in milliseconds, the inverse of the activation values for the completions were scaled up by a factor of 400. Note that this represents a claim that the activation of items in the model translates directly into reaction times. This reaction time formula is somewhat different from that used in modeling the fan effect. However, they are similar in that reaction time is a linear relationship to the inverse of the activation.

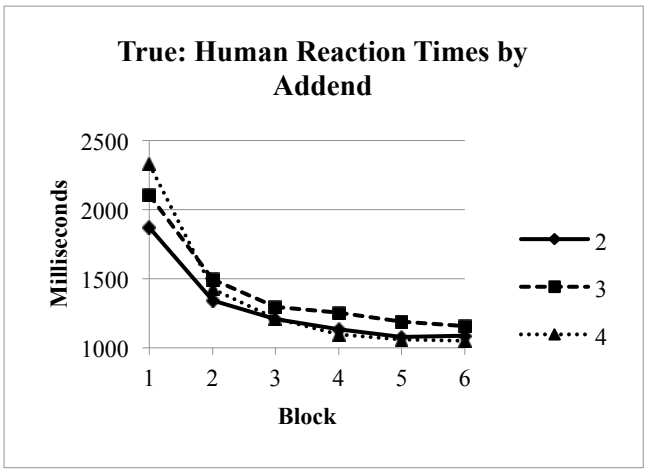

Figure 11. True reaction times for human participants in experiment 4.

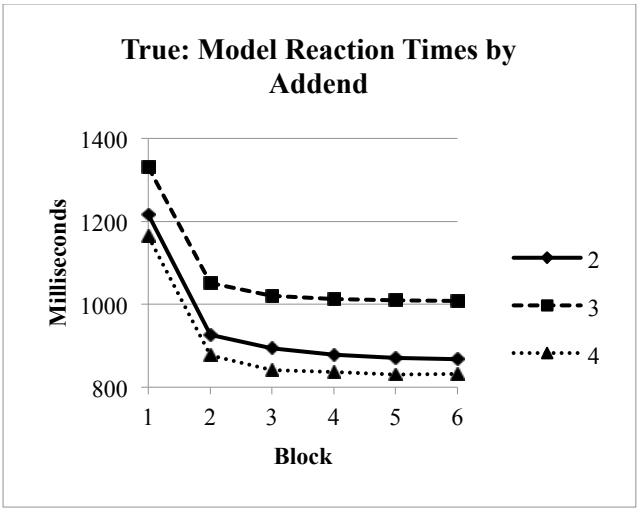

Figure 12. True reaction times for the model in experiment 4.

Figures 11 and 12 present the human and model results, respectively. Note that the reaction time for the addend 3 questions is slowest in both the human data and the simulation. Zbrodoff (1995) concluded that there were no differences between the three addend conditions. This was because she assumed that only a linear trend in the relationship between addend size and reaction time was possible, and none was found. However, the higher reaction times for the addend 3 problems were due to a real quantifiable effect, and were predictable. The reason for non-linearity was because of a fan effect pattern in the stimuli. The problems with a numerical addend of 3 had a greater fan (7) than did the problems with addends of 2 or 4 (fan of 6). Because the relationship between addend and reaction time is not linear, taking the slope of the reactions times by addend was inappropriate and occluded the results.

We can also see that model learns faster than the human participants and very quickly reaches asymptote. As noted above, this is because the model does not calculate the answers. To avoid making the graphs too small only the first six blocks are presented. However, after six blocks the human data was at asymptote.

Experiment 3

In Experiment 4, only the fan is manipulated. Experiment 3 is the same as Experiment 4 except that frequency was manipulated in the same way as in Experiment 2. That is, the questions with the smaller numerical addends were presented more frequently. The model used here was exactly the same as the one used to model Experiment 4: no parameters were altered.

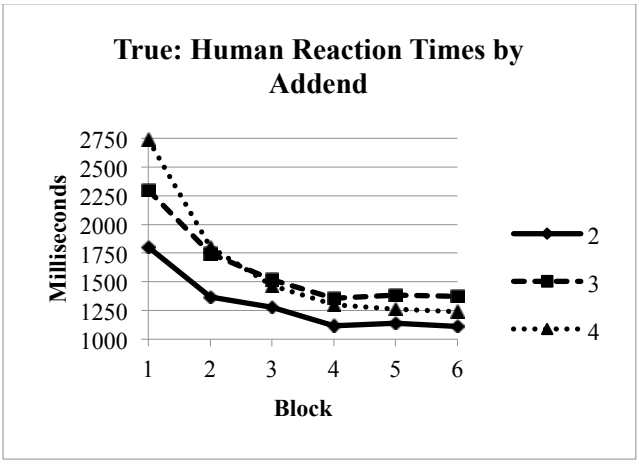

Figure 13. True reaction times for human participants in Zbrodoff's (1995) experiment 3.

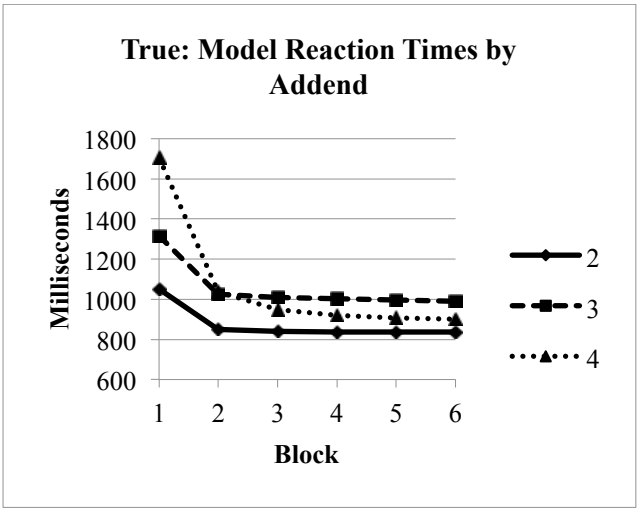

Figure 14. True reaction times for the model in Zbrodoff's (1995) experiment 3.

Figures 13 and 14 shows the human data and the simulation results. Overall, the model does a good job of accounting for the results. The most impressive aspect of this is the fact that the model predicts that the addend 4 problems ought to have the slowest recall in early blocks, due to the frequency effect, but that with time, the fan effect, described in the discussion of experiment 4 (above), takes over in later blocks causing the addend 3 problems to become the slowest. An exception to the good fit of the model occurs in the later blocks (not shown on the graphs) where the model continues to exhibit a fan effect, while in the human data the addend 4 reaction times converge with the addend 3 reactions times. A possible explanation for this is that participants were using a rehearsal strategy between sessions. If participants were recalling the questions and checking them by calculation, or rehearsing them, it could produce this effect since the addend 4 questions would be harder to recall due to the low frequency of presentation (for random recall without a cue, interference should not play a role). Therefore, the addend 4

questions would not be practiced as much. Given that the model fits well in every other respect, further tests using other sources of human data need to be done.

Paper, Rock, Scissors

By modelling Paper, Rock, Scissors (PRS) play, we provide evidence that DSHM can be used to model tasks where the truths of facts change quickly and dynamically. In order to provide some base-level of expectation for what might constitute good performance in models of PRS play, some existing models are briefly reviewed.

Perceptron Models

Simple perceptron-like neural networks can be used to model human behaviour in standard PRS games (West, 1998; West & Lebiere, 2001). The networks take sets of past opponent moves as input, and provide the choice of next move as output. The networks are constructed such that they each have an output layer of three nodes, one corresponding to each of the three play options: paper, rock, and scissors. Each has one or more groups of input nodes. Each group includes three input nodes, one for each play option. Each input node is connected to each output node, as depicted in Figure 15. The connections between nodes are assigned integer values (or, weights), which start at zero.

If a network has only a single input group, it takes only its opponent's last move as input. To determine what option to play, the network compares the connections between the node in the input group corresponding to the opponent's move and each of the output nodes. The output node attached to the connection with the greatest value determines which move the network selects (ties are decided randomly). If the network's decision results in a win, the relevant connection is rewarded by increasing its value by one. If the result is a loss, the connection is punished by reducing its value by one. Ties are treated differently by two variations of this basic

network design (West & Lebiere, 2001). 'Passive' networks treat ties as neutral events and neither reward or punish the connection values after a tie. 'Aggressive' networks punished connections leading to ties by 1. Only aggressive networks were tested in West (1998).

Networks with two or more input groups take a set of the opponent's last moves as input. Each additional input group beyond the first corresponds to a move further back in the opponent's play history. For example, with two input groups, one corresponds to the opponent's last move as input, while the other takes the opponent's second to last move as input. For these networks the output is determined by summing the connections between one node from each input group (corresponding to the move played on that past occasion) and each output node. Rewards and punishments are applied to all connections that contribute to the output decision. In addition to being labeled as either passive or aggressive, these networks were also labeled as 'lag 1', 'lag 2' or 'lag 3' depending on whether they attended to opponents' last one, two, or three past moves.

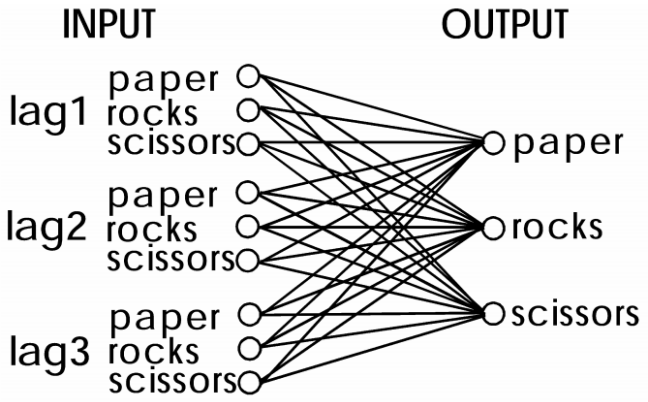

Figure 15. Perceptron networks.

Both West (1998) and West and Lebiere (2001) concluded that the aggressive lag 2 network provided the best model of human PRS play. Both the humans and the aggressive lag 2 networks were able to beat the passive lag 2 (West & Lebiere, 2001), and aggressive lag 1 networks (West, 1998; West & Lebiere, 2001), by statistically significant margins. On average, humans lost to the aggressive lag 2 networks by a small margin (West & Lebiere, 2001). However, the authors suggest that this may be due to imperfect attention and motivation on the parts of the human participants.

Perceptron Rock=2 The standard PRS game played by humans and network models in West (1998) and West and Lebiere (2001) were perfectly symmetrical. The three play options were identical in that each beat one of the other two moves and lost to the other; a rock versus scissors win was no different from a scissors versus paper win. In Rutledge-Taylor and West (2004), a modified version of PRS, where rock versus scissors wins were worth two points, while the other two outcomes were worth only one point each, was investigated. Ten human participants played one game against each of three network opponents. The network opponents were: the aggressive lag 1, the aggressive lag 2, and a 'rock=2' lag 1. The rock=2 network rewarded networks connections by two when it won with rock. Table 6 presents the mean points differences in the final scores of games between the human participants and each of the network models. All games consisted of 300 trials each.

Table 6: Humans versus networks

Network Model	Mean Pts. Diff.
Agg. Lag 1	16.5
Agg. Lag 2	5.7
Rock=2 lag 1	25.6

Two conclusions were drawn from this experiment: 1) humans were able to take advantage of the fact that wins using rock were worth two, while all three network opponents were not; 2) the rock=2 network performed the worst of all the network models due to the fact that rewarding rock wins by two became a liability (and not the anticipated advantage). This larger reward unbalanced the reward system in such a way that caused the rock=2 network played rock too frequently, and this was exploited by the human players.

An additional observation was that in Rock=2 PRS the frequencies with which player played each of the possible moves did not predict the game's final scores. For example, if two players each play paper, rock and scissors exactly 1/3 of the time each, they will tie, on average, if they are playing randomly. However, it has been demonstrated that human players (and the network models) do not play randomly (West, 1998; Rutledge-Taylor & West, 2004).

Rather than playing randomly, superior players exploit weaker opponents by predicting their opponents' moves and making winning moves. As a result, they are able to achieve higher win rates than would be predicted by play probabilities alone. This effect is particularly important in the Rock=2 game, as being able to orchestrate rock versus scissors plays and to be able to avoid the opposite play is crucial to success in this game. Thus, to evaluate performance, we use a measure called the *strategy index*.

The strategy index is calculated according to formula 1, below. Given two players, player 1's average points difference is player 1's total points minus player 2's total points, divided by the number of games played. The predicted points difference is calculated using game theory (i.e., each player is assumed to have played randomly according to probabilities determined by the actual ratios with which the options were chosen). The strategy index is the difference between these two measures. A positive strategy index indicates a superior ability to

correctly anticipate opponent's plays and achieve a higher than probabilistically predicted number of points. The raw strategy index is relative to the number of trials per game, so, it can be also represented as a percentage as in formula 2.

(1) Strategy index = average points difference per game – predicted points difference per game

(2) Strategy index percentage = Strategy index / number of trials per game

For example, if two players play a game of 300 trials, and each player plays paper 100 times, rock 100 times, and scissors 100 times, game theory predicts that they will tie (on average). However, it is possible for one player to win all 300 trials by always matching the opponent's move with the move that beats it. In this case, the winning player would score a perfect strategy index of 300 (or 100%).

ACT-R Models

ACT-R models of both standard PRS (Lebiere & West, 1999) and of Rock=2 PRS (Rutledge-Taylor & West, 2005) have been created. Both employ an exemplar based approach and manipulate noise to maximize fit to human data. For both models a chunk type with four slots is used. The *isa* slot is tagged with PRS to indicate a PRS relevant chunk. The three remaining slots encode a sequence of three moves by the model's opponent: the lag0 slot is the opponent's current or predicted move, lag1 is its previous move, and lag2 is its second to last move. An example is illustrated in Figure 16.

```
Goal
  isa   PRS
  lag2  Paper
  lag1  Rock
  lag0  nil
```

Figure 16. Example chunk

When the model's opponent makes a move, a chunk encoding the opponent's last three moves is put into the goal buffer, and then popped to make it a chunk in memory (either creating a new chunk or reinforcing an existing chunk).

To predict the opponent's next move, the model attempts to retrieve a chunk from memory that matches the opponent's last two moves (slots lag1 and lag2). The value of the lag0 slot is the move the model predicts its opponent to make. The model then plays the move that beats the opponent's predicted move.

ACT-R Rock=2 Several ACT-R models of human Rock=2 PRS play were presented in Rutledge-Taylor and West (2004). These models were similar to those appearing in Lebiere and West (1999), however, they differed in that they were designed to be sensitive to the unequal payoffs in the Rock=2 game. This sensitivity was achieved by reinforcing certain kinds of chunks more than others depending on what the opponent's last play was. Rutledge-Taylor and West (2005) tested three variations on this strategy. One model paid extra attention to cases when its opponent played rock; another attended more closely to scissors; while the third attended more closely to both rock and scissors (effectively paying less attention to paper).

The result was that the third model provided the best match to the human data. This makes intuitive sense in that a human player is likely to incorporate a defensive component to his

or her game, which is to be wary of when the opponent is likely to play Rock; however, he or she might also incorporate an offensive component which is to also focus on when the opponent might play scissors. Winning with Paper, or losing to a Paper play, is a less important event in the game.

DSHM PRS Models

Given the broad similarities between DSHM and the declarative memory system of ACT-R (Rutledge-Taylor & West, 2008), the DSHM models here were based on the ACT-R models described above.

The DSHM models took sequences of opponent's plays, encoded as ordered complex items as input. The items consisted of two, three, or four atomic items, for lag 1, lag 2 and lag 3 models respectively; the extra item is the predicted or current play by the opponent. Additionally, the λ_o of each model was set to the number of lags. The right most item represented the opponent's last move, while items to the left represented previous plays. For example, if the opponent's last few plays were:

…, rock, paper, paper, rock, scissors

Then a lag 1 DSHM model would learn the following pattern after scissors was played:

[rock:scissors]

Thus reinforcing the association between 'scissors' as a play that follows 'rock'. A lag 3 DSHM model would learn the pattern:

[paper:paper:rock:scissors]

DSHM models of two or more lags incorporate some of the learning of shorter lagged models due to the fact that they will learn all combinations of sequences of plays with lengths up to the λ_o parameter value (see calculation 8 to review how ordered items are stored). This results in a potential liability for DSHM, as shorter sequences receive repeated reinforcement for several consecutive trials. Thus, in this respect, DSHM models of PRS differ somewhat from both the perceptron-like networks, and the ACT-R models discussed above.

Rock=1 Simulations

A variety of DSHM PRS players were built. The manipulated parameters were: number of lags and the dimensionality of the vectors used to represent items. Each DSHM model played against the aggressive lag 1, aggressive lag 2, and passive lag 2 network models.

Evaluation To determine which DSHM model performed most like human players, data from West and Lebiere (2001) was used as a comparison: human players average 9.99 (s.d. 19.61) more wins than the aggressive lag 1 networks after 300 trials, lost to the aggressive lag 2 models by an average margin of 8.89 (s.d. 19.74) after an unreported number of trials, and beat the passive lag 2 by 11.14 wins after 287 trials.

Given that understanding human play against the aggressive lag 2 networks is difficult due to confounding factors discussed in West and Lebiere (2001) and the fact that an exact target win difference (after a fixed number of trials) is not available, comparison to the data against this opponent was simplified.

The DSHM models were rated according to the mean squared difference between their average final scores against the aggressive lag 1 and passive lag 2 networks, and the average final scores of humans against these models. Additionally, DSHM models that win against the aggressive lag 2 were disqualified as potential models of human play. This is because all that is certain about human performance against the aggressive lag 2 networks is that humans lost to these networks, on average.

Results Of all the models tested, one produced results that came very close to the human data. The Lag 3 DSHM model with 1024 dimensional vectors scored an average of 10.89 wins more than the aggressive lag 1 network, 13.24 more wins than the passive lag 2, and lost to the aggressive lag 2 by an average of 6.22 wins per game.

The fact that the best DSHM model was a lag 3, not a lag 2 model, was surprising at first. Lebiere and West (1999) and West and Lebiere (2001) found that lag 2 ACT-R and lag 2 network models provided the best fit to the human data. However, the fact that the DSHM lag 3 models incorporate lag 2 and lag 1 sequences means this result is not inconsistent with previous findings. The DSHM lag 3 model gives most weight to lag 1 sequences, lag 2 sequences second most, and lag 3 sequences the least weight. So, it could be argued that, on average, the lag 3 DSHM models are more like lag 2 ACT-R and network models than the lag 3 ACT-R and network models.

Rock=1 Model Comparison

The three different types of models of PRS play are compared: ACT-R, DSHM, and perceptron-like networks. In each case, the model played games consisting of 300 trials against the aggressive lag 1 network, and games of 287 trials against the passive lag 2 network. For each

model, we record the mean difference in the number of wins scored by the model and the opponent network. The sum of the squares of the differences between the model's results and the human results are presented as a basis for comparing the model's fit to the human data.

The best ACT-R model was taken from Rutledge-Taylor & West (2005). It was exemplar based, and used the following parameters: ANS=0.28, OL=NIL. The best DSHM model was the lag 3 with vectors of 1024 dimensions. The best network model was the aggressive lag 2 network.

New network versus network simulations were run for this comparison: 10000 games were run against each of the two benchmark opponents. The mean difference in wins between the aggressive lag 2 and aggressive lag 1 networks was somewhat lower than was found in West and Lebiere (2001). However, given the high standard deviation on the win differences, both the results found here and those found in West and Lebiere (2001) may be valid accounts of network play.

Table 7 summarizes the best models of human Rock=1 PRS play. The DSHM model produces the closest fit to the human data, i.e., it scored the lowest mean squared error. Additionally, the DSHM model lost to the aggressive lag 2 model by a mean win difference of 6.22, which supports this model as a good account of human PRS play.

Table 7: Models of Human Rock=1 PRS

Opponent	Human	ACT-R	DSHM	Network
Agg. lag 1	9.99	12.30	10.89	5.76
Pas. lag 2	11.14	8.15	13.24	10.44
Rating		14.27	5.22	18.37

Rock=2 Simulations

The mechanism for building a sensitivity to the unequal payoffs of the Rock=2 game in the DSHM models was essentially the same as for the ACT-R Rock=2 models. Three versions of the Rock=2 DSHM models were created: one that attended to opponents' rock plays more, another that attended to scissors more, and one that gave preference to both rock and scissors. This extra attention was achieved by training the models twice on sequences ending in these plays. For each of the three variations on the Rock=2 DSHM player, the combinations of three different lags and seven vector dimensionalities were tested.

Evaluation The results for each DSHM player were compared to the human data from Rutledge-Taylor and West (2004). A mean squared error approach was used. Six data points were compared: the mean points differences versus the three network models, and the strategy indices versus these opponents.

Results The results were somewhat predictable based on the DSHM rock=2 and ACT-R Rock=2 results: The lag 3 DSHM model with 1024 dimensional vectors that paid extra attention to both rock and scissors produced the best fit to the human data. It matched five of the six data points very well. However, this model failed to defeat the aggressive lag 2 network model. There were other DSHM models that beat the aggressive lag 2 network, but failed to match the other five data points well (e.g., the margins of victory were too great).

Rock=2 Model Comparison

As with the Rock=1 comparison, ACT-R, DSHM and perceptron-like network models of human Rock=2 PRS play are discussed. The ACT-R results are taken from Rutledge-Taylor and West (2004), the DSHM model is the one discussed above. As with the Rock=1 comparison, new network versus network data was collected. The network model of Rock=2 played each of the benchmark opponents 10000 times. Each model played 300 trial games against the three network opponents as discussed in Rutledge-Taylor and West (2004). The evaluation method for all types of models was the same as for the DSHM models discussed above.

Figures 17 and 18 summarize the evaluations of the three model types. The confidence values for the human data in Figure 17 are estimated based on the 95% confidence intervals of a regression analysis of the rate of point difference achieved by the human players against each of the three opponents.

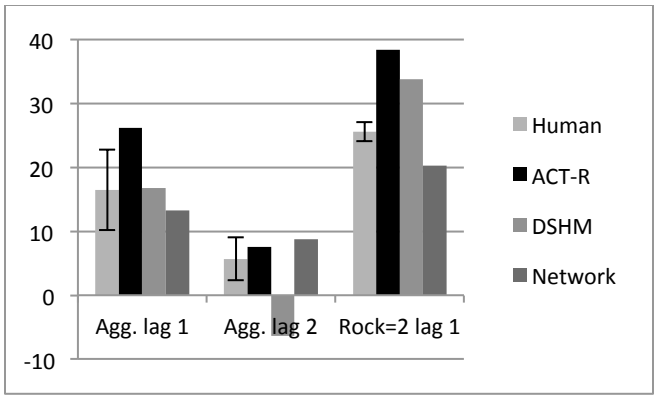

Figure 17. Points difference comparison for Rock=2 PRS. Human values include estimated 95% confidence values.

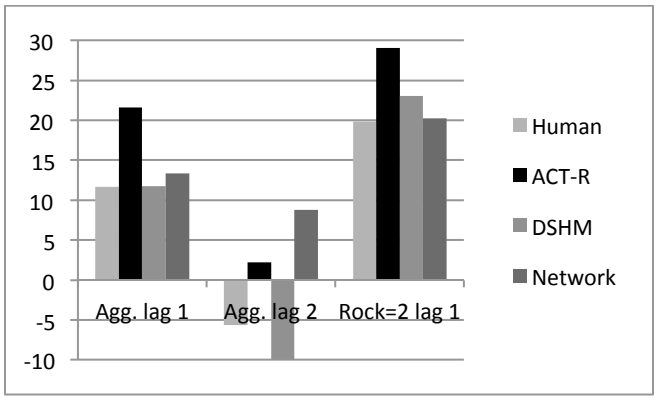

Figure 18. Strategy indices comparison for Rock=2 PRS

All three models produced good fits to the human data. However, the DSHM model is unique in that it failed to beat the aggressive lag 2 network. While human players beat the aggressive lag 2 network, human players, like DSHM, scored a negative strategy index versus this opponent. In contrast, the ACT-R and network models beat the aggressive lag 2, but did not score negative strategy indices. Thus, an obvious objective for building a superior model of human play would be to find a model that beats the aggressive lag 2, but does so despite a negative strategy index.

Learning From Delayed Feedback

Rock-paper-scissors play demonstrates that DSHM can be used to model a simple decision-making task where learning is critical. Here we apply DSHM to a binary decision-making task where the model must learn from delayed, probabilistic feedback.

While memory models can be powerful tools for detecting structure within data and making predictions from experience, memory models are generally only used to make very simple decisions, such as deciding whether a stimulus is familiar or novel in a familiarity task, or when to 'give up' trying to recall more items in a free recall task.

Conversely, cognitive architectures such as ACT-R and Soar (Laird, 2012) are designed to make complicated sequences of decisions to solve problems. ACT-R uses two systems: a procedural memory and a declarative memory. Procedural memory consists of condition-action rules weighted by utility scores. Declarative memory consists of chunks of associated data weighted by activation values.

Both the activation of a chunk in declarative memory and the utility of a condition-action rule in procedural memory can be understood as estimates of the usefulness of the chunk or rule. According to Anderson's (1991) rational analysis, the activation of a chunk in memory is an estimate of the likelihood of the information in the chunk being useful in the current situation. Likewise, a production rule's utility is an estimate of the usefulness of performing the rule's action. However, these two kinds of usefulness are quite distinct. For example, it is very useful to *know* that touching a hot object will burn you, but it is not very useful to *touch* a hot object. Thus the knowledge of the consequences of touching a hot object will have a high activation in declarative memory but the utility of touching hot objects as a rule in procedural memory will be low.

More generally, the probability of an event is distinct from the utility of the event. Thus, while computational models of human memory are well-equipped to estimate probabilities, they are not designed for decision making because they do not estimate utility.

Adapting DSHM to decision making is attractive because the complexity and intelligence of human decision making is thanks, at least in part, to the immense quantity of experiential knowledge that humans draw upon. A DSHM decision-making model may be able to scale up to decision-making tasks that require a great deal of knowledge.

In what follows, we model human performance in a sequential decision task. This task has already been modelled as a utility learning task by Walsh and Anderson (2011) using temporal difference learning models that can be incorporated into the ACT-R procedural memory. We instead model the task as a prediction task using DSHM.

Our approach to modeling decision making is related to the way DSHM models play Paper, Rock, Scissors and is conceptually similar to ACT-R models that use declarative, rather than procedural, memory to make decisions (e.g., Lebiere, Gonzalez, & Martin, 2007).

Experiment

We summarize the pertinent details of the Walsh and Anderson (2011) experimental task here. Figure 19 illustrates the structure of a single trial of the experiment. Boxes are states in the experiment. Arrows indicate transitions. Thicker arrows indicate more probable transitions. Unlabelled arrows have a probability of 100%.

In each trial of the experiment, participants make two choices that affect their chance of receiving positive feedback at the end of the trial. At the start of each trial, participants were presented with their first choice: a pair of letters, one on the left of the screen and one on the right. By pressing the corresponding arrow key, participants selected either the left or right letter.

After their selection, a cue shape flashed onto the screen for over a second. Participants then selected a second letter from a different pair of letters. After their second choice, one of two symbols appeared indicating either positive or negative feedback. This ended the trial after which a new trial would begin.

The position of a letter on either the right or left side of the screen was randomized from one trial to the next, ensuring that participants could not rely on learning a pattern of motor responses to perform the task.

Additionally, the particular letters, shapes, and symbols used in the task were randomized across participants to control for any effect of the symbols themselves.

However, for the purposes of discussing the task, we will use the letters **R** and **J** to represent the first binary choice, ▮ and ◆ to represent the pair of shapes used as cues, **T** and **V** to represent the second binary choice, and we will refer to the pair of symbols that represent positive and negative feedback as **good** and **bad**.

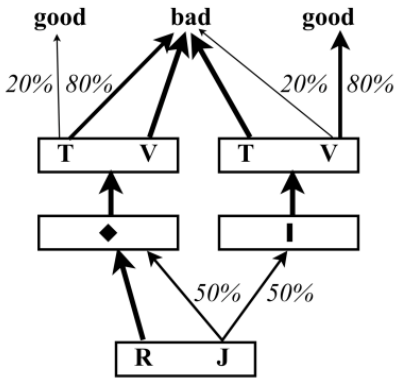

Figure 19. Experimental states and transition probabilities. Adapted from Figure 2 in Walsh & Anderson (2011).

As illustrated in Figure 19, there are three distinct decisions to be made by the participant. For each decision, there is a correct choice, that is, a choice that maximizes the chance of receiving positive feedback. Between **R** and **J**, **J** is correct. After ◆, **T** is correct, whereas after **l**, **V** is correct.

Walsh and Anderson (2011) measured participant response accuracy after 200 and 400 trials. Additionally, Walsh and Anderson used scalp-recorded event-related potential (ERP) to detect feedback-related negativity, a hypothesized neural correlate of prediction error. We attempt to model only response accuracy. We believe that prediction error, or surprise, could be modeled using DSHM, however, that is beyond the scope of this paper.

The Model

Presentation of the experimental task is simplified for the model. In this simplified format for the experiment, there are nine distinct symbols. There is a symbol that indicates the start of a trial (which we will refer to as **start**), two letters that represent the first binary choice (**R** and **J**), a pair of shapes used as cues (**l** and ◆), a second pair of letters used to represent the second binary choice (**T** and **V**) and a pair of cardinal items, **good** and **bad**, to represent positive and negative feedback.

At the beginning of the experiment, the model generates an item for each of the nine symbols used in the experiment. In the model runs discussed in this paper, each vector has1024 dimensions.

The memory vectors for each item are initialized, in a non-standard manner, to a state of optimism, such that the model initially believes that any symbol could be followed by **good**. Noise is added to the memory vectors to create random variation in the initial choices made by the simulated participants. The initialization of the memory vectors is explained in detail under

subheading *Initializing memory to a state of optimism.* The memory vectors are used to make decisions and are updated after every iteration over the course of the experiment.

The experiment is run for 400 trials. Each trial consists of five iterations as illustrated in Table 8. On the first iteration in a trial, the **start** symbol appears and the model must select **R** or **J**. On the second iteration, the selected letter appears. On the third iteration, a cue appears (either ◆ or ▌) and the model must select either **T** or **V**. On the fourth iteration, the selected letter appears. On the fifth iteration, either **good** or **bad** appears and then a new trial begins.

The model keeps track of what symbols have occurred so far in the current trial and the order in which those symbols occurred. This information can be understood as being held in a working memory buffer, but how this information is held is not a theoretical commitment of the model.

Table 8: The symbols presented to the model and choices available at each iteration in a trial.

Iteration	1	2	3	4	5
Presented	**start**	**R** or **J**	◆ or ▌	**T** or **V**	**good** or **bad**
Choices	**R** or **J**		**T** or **V**		

On each iteration of the experiment, the model updates the memory vectors corresponding to each symbol that has appeared so far in the current trial. If, on the third iteration of a trial, the symbols presented so far are "**start J** ◆", then, on that iteration, the model will update the memory vectors for **start**, **J**, and ◆. If the model selects **T**, then on the fourth iteration it will be presented the letter **T**, and the memory vectors for **start**, **J**, ◆, and **T** will be updated. Memory vectors are updated with observed relationships between symbols, as detailed in *Updating memory using open n-grams.*

Decisions are made by selecting the choice that has the greatest similarity to the probe vector. The probe vector is constructed to ask the question, "Of the choices available, which choice is more likely to be followed by **good**?". This is detailed in *Probing memory*.

Gradually, the model learns that for each choice, there is a better alternative. Learning happens for each of the three choices at roughly the same rate as human participants performing the task, as illustrated in the next section, *Comparing the model to human performance*.

Developing this model required addressing two questions that have not been previously addressed in DSHM or BEAGLE models, namely, how to model motivation and how to learn dependencies between non-consecutive events. The question of modeling motivation is addressed in *Representing positive and negative feedback* and in *Initializing memory to a state of optimism.* The question of learning dependencies between non-consecutive events is addressed in *Updating memory using open n-grams.*

Comparing the model to human performance

Walsh and Anderson (2011) had 13 participants in their experiment. Each participant performed two blocks of 400 trials. Walsh and Anderson present data aggregated across the 26 experimental blocks. Figure 20 depicts mean response accuracy with standard error indicated by error bars.

For the purposes of comparison, we average across 26 runs of the model. On each run, the model performs 400 trials. Comparing Figures 20 and 21, we see that performance of the model roughly mirrors human performance. Performance for both humans and the model is better in the latter half of the trials. Both humans and the model learn that **V** is the correct choice after **I** with relative ease, learn that **J** is a better choice than **R** more slowly, and have the most difficulty learning that **T** is the correct choice after ◆.

The reliability with which the model learns to choose **T** after ◆ varies with a parameter, the *optimism coefficient*. This parameter is explained in the section *Initializing memory to a state of optimism*. The parameter encourages exploration, but if the value is too high, the model tends to explore too much and insufficiently exploit knowledge of the task gained by this exploration. Figure 21 shows results for an *optimism coefficient* of 30.

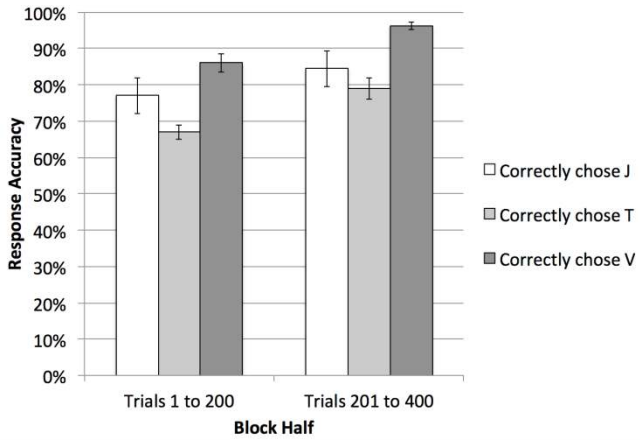

Figure 20. Response accuracy for human participants. Adapted from Figure 3 in Walsh & Anderson (2011).

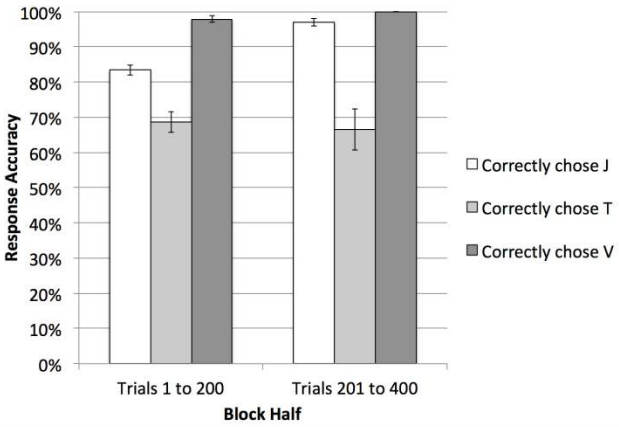

Figure 21. Response accuracy for the model.

While the model captures the overall pattern of human performance, there are two noteworthy differences. First, the model often learns to do the task nearly perfectly, whereas human participants do not tend to do so. The model's rate of correctly choosing **V** after **I** reaches ceiling in the first block half and so does not improve significantly in the second block half. However, the model's accuracy can be impaired by using vectors with far fewer dimensions. In Figure 21, we use 1024 dimensions; 256 may produce a closer fit to human performance.

Second, the model's mean rate of correctly choosing **T** after ◆ does not increase in the second block half. Improving the model's performance after ◆ may require implementing an attentional mechanism to allow the model to learn that the relationship between ◆ and **T** is critical to the feedback received and thus needs to be weighted more heavily. Implementing an attentional mechanism is discussed in *Probing memory.*

Representing positive and negative feedback

In the experimental task, participants receive positive or negative feedback at the end of each trial. Positive feedback and negative feedback take the form of arbitrary symbols that the participant has been told indicate "1 point gained" or "no points gained" respectively. Participants in Walsh and Anderson's (2011) experiment were promised $1.00 for every 50 points they gained.

Our best model makes no attempt to represent positive and negative feedback as being associated with positive or negative feelings. The model's goal is to act to maximize the likelihood of getting **good** to appear at the end of the trial. The model avoids choices that are likely to cause **bad** to appear simply because those choices are less likely to cause **good** to appear.

We have explored modeling the **bad** symbol as an aversive stimulus. This is implemented by setting the environmental vector of **bad** to be the negation of the environmental vector of **good**. We find that when **bad** is represented this way, the model becomes, as one might expect, risk averse and reluctant to explore possibilities. As a result, when **bad** is the negation of **good**, the model never learns that **T** yields a 20% chance of **good** after the cue ◆. Instead, the model chooses **V** regardless of which cue has been presented because it quickly learns that **V** often yields **good** and is reluctant to try its chances with **T**. We speculate that if humans were subjected to an aversive stimulus (e.g., an unpleasant image) every time they received negative feedback in this task, we might observe an impairment in problem solving performance similar to when **bad** is modelled as the negation of **good**. Testing this prediction of the model is a matter for future work.

Conversely, representing **bad** as a neutral stimulus encourages the model to try selecting options that have frequently yielded **bad**, which, when combined with optimism, leads to a more thorough exploration of the choices available within the task, allowing the model to better understand the consequences of its choices and discover the optimal pattern of decisions (see Figure 21).

The contrast between how the model behaves when **bad** is modelled as aversive versus neutral is consistent with *broaden-and-build theory* of positive emotions, which holds that positive emotions encourage exploration by broadening the repertoire of actions considered, whereas negative emotions narrow that repertoire (Fredrickson, 2001).

Initializing memory to a state of optimism

At the start of the experiment, memory vectors are initialized to a state of optimism. For all symbols in the task (except for **good** and **bad**), the model initially believes that symbol may later be followed by **good**.

The purpose of optimism is to motivate the model to select untried (or infrequently tried) choices. Without optimism, the model would simply repeat the same choices over and over again, as the model has no reason to believe that the other options would be any better.

Lebiere et al. (2007) also encountered this problem in modeling decision-making and likewise decided to initialize their ACT-R model to a state of optimism.

Making **bad** an aversive stimulus is an alternative to optimism that motivates the model to avoid choices that yield **bad**. However, there are two disadvantages to this approach: (1) Human participants are not punished for incorrect choices: they are rewarded for correct choices, and so such a model is performing a different task than the task that humans are asked to perform, and (2) either with or without optimism, variants of the model that are punished for incorrect choices fail to learn that **T** is the correct choice after the ◆ cue, as discussed in the previous section.

Without optimism, the memory vector for a symbol is initialized to be equal to the environmental vector for that symbol. This serves the purpose of adding random noise such that the initial decisions made by the model vary from one simulated participant (i.e., run of the model) to the next.

With optimism, an expression meaning "this is followed by **good**" is added to the memory vector when it is initialized. For any symbol *s,*

$$\mathbf{m}_s = \mathbf{e}_s + a(l(\mathbf{\Phi}) \Theta \mathbf{e}_{good}) \quad (23)$$

where a is the *optimism coefficient*, a scalar value and parameter of the model. Figure 21 shows performance of the model for $a = 30$.

Updating memory using open *n*-grams

In the experiment, the choice **J** is more likely to lead to **good** than the choice **R**. However, there are two intervening events between feedback and the initial choice: the appearance of a cue and a second decision. In human memory, is a direct association formed between the first choice and the feedback at the end of the trial? Or do people rely on a chain of associations, learning that **J** leads to a particular cue, which in turn leads to a choice, which in turn leads to an increased probability of positive feedback?

With our model we have assumed that people form a direct association between their initial choice and the feedback received at the end of the trial. The memory update scheme used is a modification of the scheme used by BEAGLE (Jones & Mewhort, 2007) and inspired by Hannagan, Dupoux, and Christophe (2011).

Hannagan et al. (2011) considered a variety of models for how letter position in words is encoded in memory and found that the encoding scheme, *unconstrained open bigrams,* provided a good model of the kinds of errors people typically make in spelling. In this scheme, all pairs of letters within a word are associated with each other, even pairs of letters that are not next to each other in the word.

In our scheme, at each iteration of a trial, all symbols presented so far in a trial are associated with each other and all groups of symbols are associated with each other, even if the symbols do not occur consecutively.

For example, on the final iteration of a trial "**start R ◆ T good**" the memory vector for **R** is updated with,

(24) $l(\mathbf{\Phi}) \Theta \mathbf{e}_{good}$ i.e., "this came before **good**"
$+ l(l(\mathbf{\Phi}) \Theta \mathbf{e}_{u}) * \mathbf{e}_{good}$ i.e., "this came before **◆ good**"
$+ l(l(\mathbf{\Phi}) \Theta \mathbf{e}_{T}) * \mathbf{e}_{good}$ i.e., "this came before **T good**"
$+ l(l(l(\mathbf{\Phi}) \Theta \mathbf{e}_{u}) * \mathbf{e}_{T}) * \mathbf{e}_{good}$ i.e., "this came before **◆ T good**"

R is also updated with "this came after **start** and came before **good**", "this came after **start** and came before **◆ good**", etc.

Probing memory

The probe vector is constructed to ask memory the question "Given what happened in the trial so far, of the choices available, which choice is more likely to be followed by **good**?". The probe vector is the sum of:

(25) $l(\mathbf{\Phi}) \Theta \mathbf{e}_{good} + l(l(\mathbf{e}_{a}) * \mathbf{\Phi}) \Theta \mathbf{e}_{good} + l(l(l(\mathbf{e}_{b}) \Theta \mathbf{e}_{a}) * \mathbf{\Phi}) \Theta \mathbf{e}_{good}$ ... etc. ...

i.e., this came before **good**; this came before **good** and after *a*; this came before **good** and after *b* and *a*; etc. where *a* is the symbol presented on this iteration, *b* is the symbol presented on the previous iteration, and so on, back to the beginning of the current trial. For the first decision in a trial, the probe vector is simply:

(26) $l(\mathbf{\Phi}) \Theta \mathbf{e}_{good} + l(l(\mathbf{e}_{start}) \Theta \mathbf{\Phi}) \Theta \mathbf{e}_{good}$

i.e, this came before **good**; this came after **start** and before **good**. For the second decision, the probe contains four terms, i.e.,

$$(27) \qquad l(\mathbf{\Phi}) \Theta \mathbf{e}_{good} + l(l(\mathbf{e}_{\blacklozenge \, or \, \blacksquare}) \Theta \mathbf{\Phi}) \Theta \mathbf{e}_{good} + l(l(l(\mathbf{e}_{\mathbf{R} \, or \, \mathbf{J}}) \Theta \mathbf{e}_{\blacklozenge \, or \, \blacksquare}) \Theta \mathbf{\Phi}) \Theta \mathbf{e}_{good} + l(l(l(l(\mathbf{e}_{start}) \Theta \mathbf{e}_{\mathbf{R} \, or \, \mathbf{J}}) \Theta \mathbf{e}_{\blacklozenge \, or \, \blacksquare}) \Theta \mathbf{\Phi}) \Theta \mathbf{e}_{good}$$

Decisions are made using *resonance*, that is, by measuring the similarity, as calculated by vector cosine, between the probe vector and the memory vectors of the two available choices. The model selects as its choice the symbol corresponding to the memory vector with the highest similarity to the probe is selected by the model as its choice.

Note that for the purposes of this particular model, the only symbols that need to have memory vectors are the choice symbols, represented here as **R**, **J**, **T**, and **V**. However, it is a theoretical or architectural commitment of DSHM that all perceptual stimuli have an associated memory vector.

The probe vector is constructed to ask a distinct question for each term that comprises it. For the first decision, those questions are "Which of **R** or **J** is most likely to lead to **good**?" and "Given that the **start** symbol was just seen, which of **R** or **J** is most likely to lead to **good**?". For the second decision, four questions are posed by the probe: "Which of **T** or **V** is most likely to lead to **good**?", "Given the cue shape that was just seen, which of **T** or **V** is most likely to lead to **good**?", "Given the cue shape that was just seen and the previous decision that was made, which of **T** or **V** is most likely to lead to **good**?", and "Given the cue shape that was just seen, the previous decision that was made, and the fact that the start symbol was seen before that, which of **T** or **V** is most likely to lead to **good**?". The similarity of the probe vector to a particular choice's

memory vector can be thought of as a mean of the likelihoods of that choice for each of those questions.

For example, in this experiment, **V** is, overall, very likely to be followed by **good**, but if **V** is preceded by the ◆ cue or the choice **R**, **V** will not be followed by **good**. The one question in the probe that is answered with "very likely" for **V** is answered with "not likely at all" by the other three questions, and as a result, when preceded with the ◆ cue, the model rightly estimates that the chances of **V** being followed **good** is very low and as a result the model may correctly select **T** in those circumstances instead.

One could introduce into the model an attentional mechanism that weights the terms of the probe. The appropriate value of these weights could be gradually learned by the model as it discovers which relationships are reliable and informative predictors and which relationships are not reliable or not useful predictors. Such a mechanism has not been implemented in this model.

Walsh and Anderson (2011) informed participants before the task "that the initial choice affected which cue appeared, and that the final choice depended only on the cue that appeared". Given this information, one might assume that participants only attended to two relationships:

1. The relationship between the initial choice, the cue shape that appears, and the feedback at the end of a trial, and
2. The relationship between the cue, the final choice, and the feedback received.

In our model, we instead make the following assumptions:

1. Participants may not have fully absorbed the clues provided by the instructions,
2. Attention and recall is, to a degree, automatic such that participants will pay attention to and be affected by things that they may know are irrelevant to the task, and that

3. The rapid pace of the experiment prevents participants from reasoning particularly carefully about the task and the decisions they are making.

Given these assumptions, we believe our model is a reasonable approximation to participant behaviour.

Discussion

We hold that, in order to bridge the gap between human experience and neural connectivity, explanations at both the symbolic and sub-symbolic levels of description are necessary parts of theory in cognitive science. As we illustrate in this paper, cognitive models that use vector-symbolic architectures intrinsically operate at both of these levels of description and thereby provide a needed bridge between the two kinds of explanation.

The vector-symbolic architecture itself is at the sub-symbolic level. The linear algebra operations on vectors comprise the architecture: addition, circular convolution, and permutation as tools for constructing representations; circular correlation, inverse permutation, and vector cosine as means of recovering information from those representations. All of these operations are amenable to neural implementation, as demonstrated by Eliasmith (2013).

At the symbolic level, we have the cognitive model itself, and the cognitive processes of storage and retrieval that define it. DSHM adopts and adapts the learning mechanisms of the BEAGLE model of language learning for new uses. DSHM can be applied to long-term learning, modeling biological memory systems that contain very large quantities of data, as demonstrated by BEAGLE (Jones & Mewhort, 2007) and supported by other work not reported here that uses DSHM as a recommender system for movies or research papers (Rutledge-Taylor, Vellino, & West, 2008). DSHM is capable of modeling short-term learning and unlearning, as demonstrated

by the Paper, Rock, Scissors model presented in this paper. This also suggests that long-term and short-term memory in humans may rely on the same basic mechanisms.

By virtue of its vector-symbolic representations, DSHM easily accounts for basic memory phenomena such as interference and the frequency effect, as demonstrated by the DSHM models of the fan effect and the problem-size effect. The success of DSHM as a model of the delayed-feedback learning task demonstrates that DSHM can be applied to decision-making tasks that have been typically understood as utility learning tasks handled by a distinct procedural memory system. Indeed, converging evidence from the modeling literature suggests that procedural and declarative memory may not be so different (Humphreys, Bain, & Pike, 1989; Jamieson, Crump, & Hannah, 2012; Jamieson & Mewhort, 2011).

Simon (1969) famously argued that to account for the complexity of human behavior we need to account for the influence of two environments: the external environment of the world and the internal environment of memory. Dynamically structured holographic memory provides an account of memory that can, as we have argued, be implemented as a realistic neural model, can account for a wide variety of experimental memory phenomena, and can be scaled up to long-term learning. Thus we feel that DSHM is an important step towards a working model of the human mind.

References

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

Anderson, J. R. (1991). The place of cognitive architectures in a rational analysis. In K. Van Len (Ed.), *Architectures for Intelligence*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Lawrence Erlbaum Associates.

Anderson, J. R. & Reder, L. R. (1999). The fan effect: New results and new theories. *Journal of Experimental Psychology: General, 128(2),* 186-197.

Bunting, M. F., Conway A. R. A., & Heitz, R. P. (2004). Individual differences in the fan effect and working memory capacity. *Journal of Memory and Language, 51(4),* 604-622.

Cox, G. E., Kachergis, G., Recchia, G., & Jones, M. N. (2011). Towards a Scalable Holographic Representation of Word Form. *Behavior Research Methods, 43*, 602-615.

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

Eliasmith, C., & Thagard, P. (2001). Integrating structure and meaning: a distributed model of analogical mapping. *Cognitive Science, 25*, 245-286. doi: 10.1016/S0364-0213(01)00036-2

Franklin, D. R. J., & Mewhort, D. J. K. (2013). Control Processes in Free Recall. In R. West & T. Stewart (eds.), *Proceedings of the 12th International Conference on Cognitive Modeling* (pp. 47-52), Ottawa: Carleton University.

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 Slezak, P. (Ed.), *Proceedings of the Joint International Conference on Cognitive Science* (pp. 133-138). Sydney, Australia: University of New South Wales.

Goetz, P. & Walters, D. (2000). A neuronal basis for the fan effect. *Cognitive Science, 24(1)*, 151-167.

Hamann, M. & Ashcraft, M. (1986). Textbook presentations of the basic addition facts. *Cognition and Instruction, 3*, 173-192.

Hannagan, T., Dupoux, E., & Christophe, A. (2011). Holographic string encoding. *Cognitive Science, 35*, 79-118.

Humphreys, M. S., Bain, J. D., & Pike, R. (1989). Different ways to cue a coherent memory system: A theory for episodic, semantic, and procedural tasks. *Psychological Review, 96*, 208-233. doi: 10.1037/0033-295X.96.2.208

Jamieson, R. K., Crump, M. J. C., & Hannah, S. D. (2012). An instance theory of associative learning. *Learning & Behavior, 40*, 61-82.

Jamieson, R. K., & Mewhort, D. J. K. (2009). Applying an exemplar model to the serial reaction time task: Anticipating from experience. *Quarterly Journal of Experimental Psychology, 62*, 1757-1783.

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

Kelly, M. A., Blostein, D., & Mewhort, D. J. K. (2013). Encoding structure in holographic reduced representations. *Canadian Journal of Experimental Psychology, 67*, 79-93. doi:10.1037/a0030301

Kelly, M. A., Mewhort, D. J. K., & West, R. L. (2014). The memory tesseract: Distributed MINERVA and the unification of memory. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*. Quebec City, Canada: Cognitive Science Society.

Laird, J. E. (2012). T*he Soar Cognitive Architecture*. MIT Press, Cambridge, MA.

Lebiere, C., Gonzalez, C., & Martin M. (2007). Instance-based decision making model of repeated binary choice. *Proc. of the 12th International Conference on Cognitive Modeling,* 67-72.

Lebiere, C. & West, R. L. (1999) A dynamic ACT-R model of simple games. In *Proceedings of the 21st Annual Conference of the Cognitive Science Society*. 296-301. Simon Fraser University: Vancouver, Canada.

Murdock, B. B. (1982). A theory for the storage and retrieval of item and associative information. *Psychological Review, 89*, 609–626. doi: 10.1037/0033-295X.89.6.609

Murdock, B. B. (1993). TODAM2: a model for the storage and retrieval of item, associative and serial-order information. *Psychological Review, 100*, 183–203. doi: 10.1037/0033-295X.100.2.183

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

Radvansky, G. A., Spieler, D. H., & Zacks, R. T. (1993). Mental model organizaton. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 19*, 95-114.

Rutledge-Taylor, M. F., Pyke, A. A., West, R. L. & Lang, H. (2010) Modeling a three term fan effect. In *Proceedings of the Tenth International Conference on Cognitive Modeling*. Philadelphia, PA: Drexel University.

Rutledge-Taylor, M. F., Vellino, A., & West, R. L. (2008). A holographic associative memory recommender system, *Proc. of the Third International Conference on Digital Information Management*, 87-92.

Rutledge-Taylor, M. F. & West, R. L. (2004) Cognitive modeling versus game theory: Why cognition matters. In *Proceedings of the Sixth International Conference on Cognitive Modeling*, 255-260. Pittsburgh, PA: Carnegie Mellon University/University of Pittsburgh.

Rutledge-Taylor, M. F. & West, R. L. (2005) ACT-R versus neural networks in rock=2 paper, rock, scissors. In *Proceedings of the Twelfth Annual ACT-R Workshop*, 19-23. Trieste, Italy: Universita degli Studi di Trieste.

Rutledge-Taylor, M. F., & West R. L. (2008). Modeling the fan-effect using dynamically structured holographic memory. *Proc. of the 30th Annual Conference of the Cognitive Science Society*, 385-390.

Siegler, R. (1987). The perils of averaging data over strategies: An example from children's addition. *Journal of Experimental Psychology: General, 116*, 250-264.

Simon, H. A. (1969). The psychology of thinking: Embedding artifice in nature (Chapter 2). In *Sciences of the artificial*, 23-26. MIT Press.

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

West, R. L. (1998) Zero sum games as distributed cognitive systems. In *Proceedings of the Complex Games Workshop*. Tsukuba, Japan: Electrotechnical Laboratory Machine Inference Group.

West, R. L., & Lebiere, C. (2001). Simple games as dynamic, coupled systems: Randomness and

other emergent properties. *Cognitive Systems Research, 1*, 221-239.

Verguts, T. & Fias, W. (2005). Interacting neighbours: A connectionist model of retrieval in single-digit multiplication. *Memory & Cognition, 22*, 1-16.

Zbrodoff, N. J. (1995). Why is 9 + 7 harder than 2 + 3? Strength and interference as explanations of the problem-size effect. *Memory and Cognition, 23(6)*, 689–700.

Zbrodoff, N. J. & Logan, G. D. (2005). What everyone finds: The problem-size effect. In J. I. D. Campbell (Ed.), *Handbook of mathematical cognition* (pp. 331–346). New York: Psychology Press.