Encoding and Retrieval Efficiency of Episodic Data in a Modified Sparse Distributed Memory System

Sidney K. D'Mello (sdmello@memphis.edu)
Computer Science Department & The Institute for Intelligent Systems,
365 Innovation Drive, Memphis, TN 38152, USA

Uma Ramamurthy (urmmrthy@memphis.edu)
The Institute for Intelligent Systems, 365 Innovation Drive
Memphis, TN 38152, USA

Stan Franklin (franklin@memphis.edu)
Computer Science Department & The Institute for Intelligent Systems,
365 Innovation Drive, Memphis, TN 38152, USA

Abstract

This paper presents detailed simulation results on a modified Sparse Distributed Memory (SDM) system. We have modified Kanerva's original SDM system into an architecture with a ternary memory space. This enables the memory to be used as a Transient Episodic Memory (TEM) in cognitive software agents. TEM is a memory with high specificity and low retention, used for events having features of a particular time and place. Our earlier work focused on perfunctory, proof of concept assessments on the modified SDM system. This paper presents a detailed experimental evaluation of the modified SDM system with regard to its ability to store and retrieve episodic information.

Introduction

Episodic memory is for events having features of a particular time and place (Baddeley et al, 2001). This memory system is associative in nature and content-addressable. It has been proposed that working memory probably includes an episodic buffer that can hold episodic information for a short duration (Baddeley, 2000).

Humans have a content-addressable, associative, transient episodic memory with a decay rate measured in hours (Conway, 2001). Humans are able to recall in great detail events of the current day – where they park their cars, whom they met that morning, what they discussed, what they had for meals, etc. These details of the events/episodes stay with us only for short durations – for some hours. We hypothesize that for cognitive agents to recall such details of episodes while they interact with and adapt to their dynamic environments, they need a transient episodic memory (TEM). The Intelligent Distribution Agent (IDA) is one such cognitive software agent endowed with a TEM (Baars, & Franklin. 2003, Franklin et al in review).

IDA is a cognitive software agent (Franklin, 1997) developed for the U.S. Navy. At the end of each sailor's tour of duty, he or she is assigned to a new billet by a person called a *detailer*. IDA's task is to facilitate this process by completely automating the role of a detailer. The IDA technology (Franklin, 2001) has a number of different memory systems, including working memory, transient episodic memory, and autobiographical/declarative memory (Ramamurthy, D'Mello, & Franklin, 2003). We hypothesize that information stored in TEM, which has not decayed away, is consolidated into declarative memory at certain intervals.

Transient episodic and declarative memories have distributed representations in IDA. There is evidence that this is also the case in animal nervous systems. Some of these memory models are motivated by Sparse Distributed Memory (Kanerva, 1988). This is reasonable due to several similarities between SDM and human memory systems such as *knowing that one knows, tip-of-the-tongue effect, rehearsal, momentary feelings of familiarity, and interference.* The focus of this paper is on a modified SDM architecture that promises to be a good candidate for use as a TEM in software agents such as IDA.

Sparse Distributed Memory

SDM implements a content-addressable random access memory. Its address space is in the order of 2^{1000}. Of this space, you choose a manageable, uniform random sample, say 2^{20}, of allowable locations. These are called hard locations. Thus the hard locations are sparse in this address space. Many hard locations participate in storing and retrieving of any datum, resulting in the distributed nature of this architecture. Hamming distance is used to measure the distance between any two points in this memory space.

Each hard location is a bit vector of length 1000, storing data in 1000 counters with a range of -40 to 40. Each datum to be written to SDM is a bit vector of length 1000. Writing 1 to a counter results in incrementing the counter, while writing a 0 decrements the counter. To write in this memory architecture, you select an access sphere centered at location X. So, to write a datum to X, you simply write to all the hard locations (typically 1000 of them) within X's access sphere. This results in distributed storage. This also naturally provides for memory rehearsal – a memory trace being rehearsed can be written many times and each time to about 1000 locations.

Similar to writing, retrieving from SDM involves the same concept of access sphere – you read all the hard

locations within the access sphere of location Y, pool the bit vectors read from all these hard locations and let each of the k^{th} bits of those locations participate in a majority vote for the k^{th} bit of Y. Effectively, you reconstruct the memory trace in every retrieval operation. Effectively, the read data at Y is an aggregate of all data that have been written to the hard locations within Y's access sphere, but may not be any of them exactly.

Furthermore, this memory can be cued with noisy versions of the original memory trace. To accomplish this, you employ iterated reading – first read at Y to obtain the bit vector, Y1. Next read at Y1 to obtain the bit vector Y2. Next read at Y2 to obtain the bit vector, Y3. If this sequence of reads converges to Y', then Y' is the result of iterated reading at Y.

The Modified SDM system

Our experimental evaluation of Kanerva's original SDM for cognitive agents such as IDA that encode text based episodic data, indicated the need for an architecture modification. Episodic data refers to patterns with features of what, where, and when. Preliminary investigations that assessed SDM's ability to encode text based episodic data revealed two fundamental shortcomings. When events are unfolding, the feature vector is not always complete. So, more often, the agent has to store partial feature sets. Similarly, when the agent cues its memory for retrieval, the retrieval cues are often partial feature-sets. SDM has no generic mechanism to handle partiality in the stored patterns as well as in the retrieval cues. It considers missing features to be random noise, thereby severely effecting performance.

The second major problem with SDM is its inability to handle text. Since SDM operates in a Boolean space, encoding text requires binary representations of characters. A simple way to enforce this mapping is by encoding the ASCII representation of characters. For example, the feature "dog", would be represented as "01100100 01101111 01100111". Since interference from related features effects the retrieved trace, error in recall is introduced. During the recall procedure, if the second bit of each character in the binary representation of dog is flipped, the resultant binary patterns is "0*0*1001000 0*0*1011110 0*0*1001110". Converting this recalled binary pattern into text would result in "$/^", which at the character level bears absolutely no similarity to "dog." This example shows that a 12.5% error in the retrieval process can completely distort the feature.

The modified SDM system (Ramamurthy et al, 2004) alleviates several of the shortcomings identified with using SDM as a computational model for TEM. The modification includes migrating to a ternary memory space while maintaining a binary address space for the hard locations. Adding "don't cares" (*'s) to the 0's and 1's of the binary space yields a ternary memory space. This accommodates flexible cuing with fewer features than the actual memory trace where missing features are represented by "don't cares" (*). An adjustment was made to Hamming distance calculations such that the distance between a "don't care" (*) and a 0 or 1 was set to (0.5).

This modification to the memory space also addresses two essential features of episodic memory systems (Shastri, 2002). Episodic memory systems must have binding-error detectors and binding-error integrators. Given an event, the episodic memory trace must respond not only to partial cues, it should also be capable of distinguishing a memorized event from very similar events.

Previously simulated experiments revealed that the modified SDM system produces a significant performance improvement over the original SDM system (Ramamurthy, et al, 2004). However, the extent of the improvement was not formally quantified. Furthermore, several system parameters that maximize performance were not investigated. Hence, the experiments presented here attempt to systematically assess performance of the modified SDM system's ability to encode and retrieve episodic information with variable system parameters. These include upper and lower bounds on the degree of partiality in the encoded patterns and the retrieval cues and the size of the pattern set.

Experimental Analysis & Results

The primary focus of the experiments was to compare performance of the modified and the original SDM regarding its ability to encode episodic data. Therefore, the experiments evaluated performance of the two memory systems when patterns encoded into the memory had an increasing degree of partiality (missing features in the pattern). Retrieval was tested with fully specified read-cues and partial read-cues (binding-integration). Performance of the modified SDM system when presented with binding-errors in the retrieval cues have been investigated elsewhere (Ramamurthy et al, 2004).

The experiments also evaluated whether the modified SDM reduces some of SDM's capacity problems. SDM has been criticized for its relatively low memory capacity (Keeler, 1988). We evaluated memory capacity by testing storage and retrieval performance with pattern sets that fill the memory to approximately 25%, 50%, 75%, and 100% capacity.

Experimental Setup

The experiments were conducted by randomly initializing 100 memories. A memory in this context refers to a fully initialized SDM simulation. All performance results were averaged over these 100 memories. This approach controls for any bias in the results that may be introduced by the random memory initialization. All experiments were run on both the modified as well as the original SDM systems. Tests were conducted at 4 different memory capacity levels (C-25 ... C-100): namely 25%, 50%, 75%, and 100% (at capacity).

Memory Architecture

Each memory was randomly initialized with 10,000 hard locations. A larger sample of hard locations was avoided due to computational limitations. The capacity of a 10,000 hard-location-memory, when cued with the exact addresses of the stored patterns, is approximately 1000 patterns, since

the capacity of SDM is estimated to be 10% of the number of hard locations (Kanerva, 1988). However, when the memory is cued with noisy versions of the stored patterns its capacity greatly decreases to about 1-5% of the number of hard locations (Kanerva, 1993). Since our experiments involve encoding episodic data, the stored patterns as well as the retrieval cues had high partiality. Hence, an estimate of capacity based on episodic data was taken to be 1% of the number of hard locations. Therefore, in order to fill the memory to 25% capacity, 24 episodes were encoded; to fill memory to 50% capacity, 48 episodes were encoded, etc. Further, a model for TEM does not enforce stringent capacity requirements due to TEM's low retention, which can be enforced via an appropriate decay mechanism.

The dimensionality of the memory space was set at 448, based on the case-grammar template (Fillmore, 1968) used in the experiments. The case-grammar template selected is illustrated in Figure 1, with examples of fully specified feature-sets and partially specified feature-sets representing patterns used for encoding and retrieval.

Pattern Selection

Tests at each capacity level included 6 memory write sets (W0 … W5), each with an increasing degree of partiality in the patterns. Patterns in set W0 were fully specified (no "don't cares"), while patterns in set W5 had 5 "don't cares" (62.5% partiality). "Nathan accepts * Michael venture scheme eatery *" is an example of a partial memory-write for the W2 category (2 "don't cares"). Here, the *recipient-adjective* and the *time* features have been replaced with "don't cares" (*). The number of patterns within each write set was specified by the level of capacity being tested.

Retrieval on each memory write set (W0 ... W5) was tested by 4 read-cue sets (R0 ... R3), each with an increasing degree of missing features in addition to the missing features of the write sets. As an example, consider a 62.5 % partial read-cue: "Nathan * * * venture scheme * *". This is a R3 read-cue as it contains three missing features in addition to the two missing features in the encoded pattern shown above. The number of patterns within each read-cue set was the same as the write set that it is being tested on.

TEMPLATE:

"Agent | Verb | Recipient-Adjective | Recipient | Object-Adjective | Object | Place | Time"

EXAMPLES:

(1) "Richard | drives | joyful | Vanessa | lively | comedy | Theatre | Friday"

(2) "Richard | drives | * | * | lively | comedy | * | Friday"

(3) "Michael | answers | cousin | Nathan | nervous | queries | eatery | Tuesday"

(4) " * | answers | * | Nathan | * | queries | * | Tuesday"

Figure 1: Case-grammar template

Results & Discussion

The encoding efficiency of the Modified SDM system was compared to the original SDM system by assessing its pattern distribution and interference reduction abilities. Retrieval performance was estimated by assessing the convergence rate (retrieval rate), the quality of retrieval, and by a novel performance metric.

According to the experimental design, retrieval of every pattern in each write set (W0 ... W5) was evaluated by 4 different read cue sets (R0 ... R4). Due to space constraints, the retrieval results presented below have been averaged over these four different read cue sets.

Pattern Distribution

The distributed nature of SDM's architecture requires that a pattern should be encoded to approximately 1% of the hard locations. Although, such a distributed representation is beneficial in terms of its ability to handle partial failure, it introduces severe interference problems. Interference refers to large overlaps between the access spheres of related patterns. This is due to the hard locations being randomly initialized, while the encoded patterns are not evenly distributed and tend to cluster in the memory space. To account for this phenomenon, we introduced a simple measure to assess the distribution of a set of patterns in the memory space called the activity. A hard location is said to be active if it is involved in encoding at least one pattern. The activity of a memory is simply the percentage of its hard locations that are active. Therefore, as the memory is filled to its capacity, a good distribution should demand that its activity should proportionally increase. Figure 2 presents a comparison of the activity of the original and modified SDM as the memory was filled to capacity.

From Figure 2, we see the activities of both memories increased as the size of the pattern set increased (analogous to the memory being filled to capacity). The activity of the modified SDM was on an average 5.19% higher than the original SDM. This indicates that the modified SDM was more effective in distributing patterns with partial features even as the memory was filled to capacity.

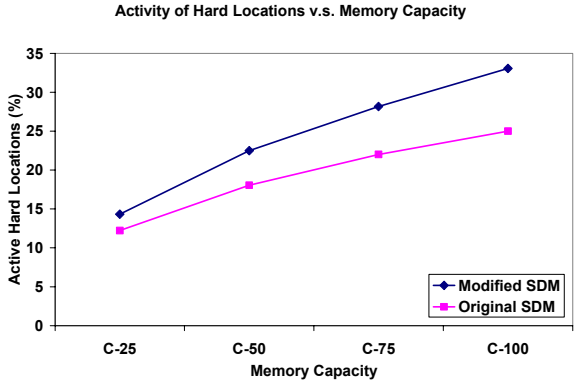

Figure 2: Pattern distributions of both memories as the size of the pattern sets increases

In Figure 3 we averaged over the different capacity levels and evaluated the modified SDM's sensitivity to partiality in the encoded patterns.

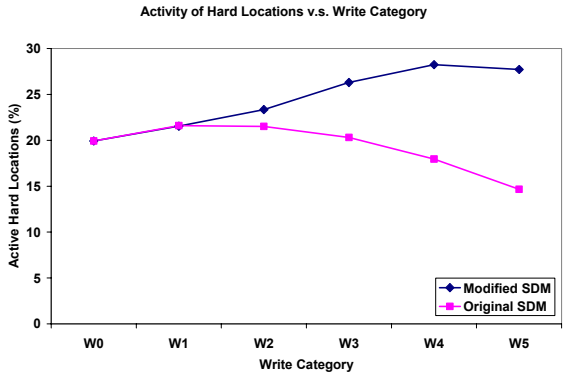

Figure 3: Pattern distribution of both memories as the degree of partiality in the stored patterns increases

From Figure 3, we see that as the partiality of the encoded patterns increased, the modified SDM performed significantly better than the original SDM. Overall, the modified SDM responded to an increase in the partiality of the encoded patterns with a 39.09% net growth in its activity, while the original SDM reported a net drop of 26.39%.

It is interesting to note that even when the memory was full (C-100), only 33.05% and 25.01% of the hard locations in the modified and original SDM respectively were active (Figure 2). This implies a clustering of the patterns in about a third of the memory space. These results are consistent with the notion of SDM's performance failures for handling non-random data (Hely, Willshaw , & Hayes, 1997) and in some sense are a justification for a domain based initialization approach as opposed to the conventional random initialization utilized in these experiments.

Convergence Rate

Convergence occurs when the distance between the read cue and the target pattern is below a threshold distance (the critical distance (Kanerva 1988), and the iterated read trace settles on a fixed point. Divergence occurs if any of the above two conditions established for convergence are violated. Hence, the *convergence rate* of a memory can be defined as the ratio of the number of times it converges to the number of retrieval operations. Figure 4 presents the convergence rates of the two memories as the degree of partiality in the written patterns increased.

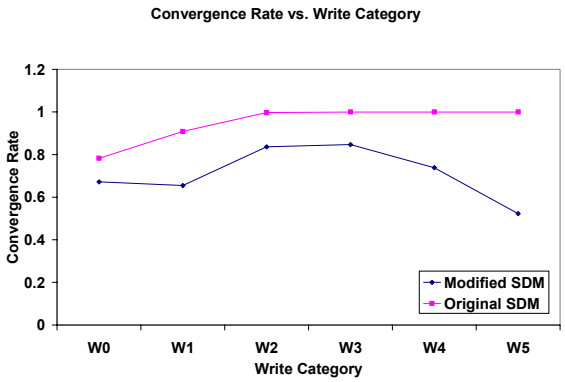

Figure 4: Convergence rate of both memories as the degree of partiality in the stored patterns increases

From Figure 4, it is clear that the original SDM converged more rapidly than the modified SDM. Its convergence rates gradually grew as the level of partiality in the encoded patterns increased, with a 27.71% net growth over the W0-W5 interval. The modified SDM showed about a 22.17% drop in its convergence rates over the same write interval. It first showed an initial drop in the convergence rate during the W0 to W1 transition, followed by a sharp jump from W1 to W2, a small increase from W2 to W3, and finally two sharp drops (W3 to W4 and W4 to W5). Intuitively, this suggests that between the W1 to W3 range, partiality in the encoded patterns actually effected greater convergence in the modified SDM system.

A closer look at Figure 4, shows that within the W2-W5 range, the original SDM reports almost 100% convergence. However, considering the highly partial read cues used for retrieval and the modest convergence rates reported by the modified SDM, one would suspect that the patterns retrieved by the original SDM are false positives. This suspicion is realized by assessing the quality of retrieval of both memory systems.

Quality of Retrieval

Convergence rate by itself can be misleading because the retrieval content is ignored. Having a very high convergence rate with low retrieval quality is equivalent to a false positive. Therefore, we coupled retrieval content with convergence rates in order to quantify retrieval quality. The quality of retrieval of a memory is the ratio of the traces perfectly retrieved to the frequency of its convergence. Figure 5 compares the quality of retrieval of both memories along the different write categories.

We see that the modified SDM significantly outperformed the original SDM across the different write categories. Retrieval quality of the original SDM gradually dropped to zero on the advent of partiality in the encoded patterns with a 99.76% net drop over the W0 to W5 range. The modified SDM exhibited a net growth of 127.01% over the same range indicating that the memory really "knows what it knows".

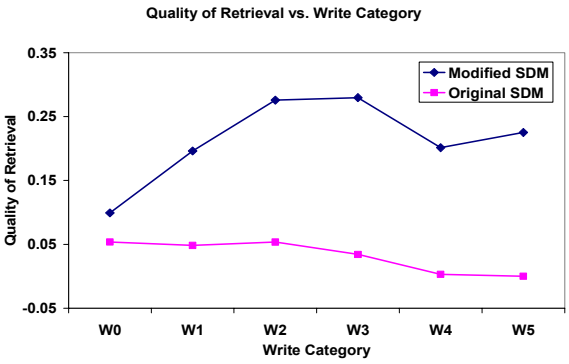

Figure 5: Quality of retrieval of both memories as the degree of partiality in the stored patterns increases

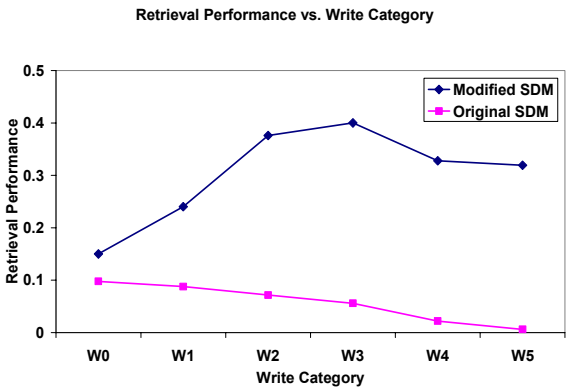

Figure 6: Retrieval performance of both memories as the degree of partiality in the stored patterns increases

Retrieval Performance

The convergence rate and the quality of retrieval metrics are useful for a low level retrieval analysis. However, they do not provide a global view into the retrieval process. Additionally, one of the problems with the quality of the retrieval metric is that it requires perfect retrieval. It can be argued that perfect retrieval is not a fundamental requirement, if a level of post-processing by virtue of approximate string matching algorithms (Knuth, 1977) is employed. Therefore, the tests were evaluated using a metric that is based on the number of features fully recovered in the retrieval process. A feature was considered to be fully retrievable if no more than two of its characters were incorrect and hence could be recovered fully with post processing. The scoring of the retrieved episodes was based on the following scale: (1) All features fully retrieved were scored as 1.0; (2) All but one feature fully retrieved were scored as 0.75; (3) All but two features fully retrieved were scored as 0.5; (4) Retrievals that were incorrectly retrieved, with more than two irretrievable features, or diverged reads were scored as 0. Figure 6 illustrates the retrieval performance of the modified and original SDM systems across the different write categories.

Figure 6 shows that the performance of the original SDM dwindled with an increase in partiality of the write patterns. Its performance curve was loosely linear with a net drop of 42.93% over the W0 to W5 range. The corresponding performance curve of the modified SDM exhibited an interesting behavior. It showed a steady 150.67% growth from W0 to W2, a smaller rise to a peak at W3, followed by a gradual drop from W3 to W5. This indicates that extension of the content-space to include the "don't cares" (*) provided a significant improvement as the percentage of missing features in the memory writes increased to a reasonable degree. Hence, doing partial writes is advantageous as "don't cares" affect more rapid convergences due to the modification to the Hamming distance calculation.

Memory Capacity

All retrieval results presented so far have been averaged over the sizes of different pattern sets. As stated above, the size of a pattern set is analogous to memory capacity. Figure 7 shows four curves, each representing the performance of the modified SDM when the memory was filled to approximately 25%, 50%, 75%, and 100% capacity.

We see that a moderate increase in the partiality of the encoded patterns improved performance. This moderate increase lies somewhere in the W1-W3 range (12.5% to 62.5% partiality). An increase in partiality at this point affects performance.

The original SDM showed a zero performance score for all experiments in which the memory was filled above 50% of its capacity (C-75, C-100), irrespective of the partiality in the encoded patterns as well as the retrieval cues. Therefore, we can claim that the modifications proposed to the SDM system definitely reduced some of its capacity problems.

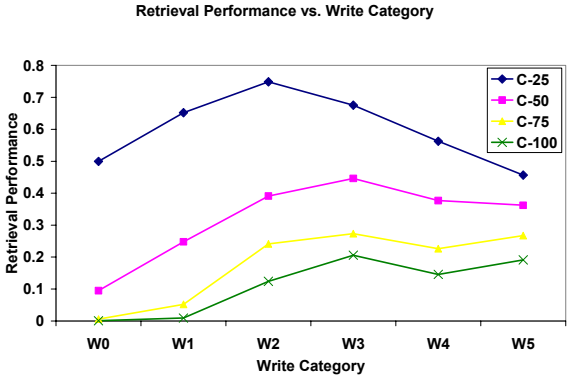

Figure 7: Retrieval performance of the modified SDM system for four different pattern sets as the partiality in the stored patterns increases

Discussion

The modified SDM system used as a TEM promises a significant performance improvement over the original SDM system. The extensive experimental simulations revealed that in all cases the modified SDM system

outperformed the original SDM system. It constrains interference by efficiently distributing the encoded patterns across the hard locations in the memory space. Its abilities in encoding partial patterns and retrieving with partial cues are also significantly better than the original SDM. Interestingly, a reasonable degree of "don't cares" in the patterns improves performance as they act as attractor basins due to the modification to the Hamming distance calculation. Finally, the modified SDM system also alleviates some of the problems related to text encoding demonstrated by its improved retrieval quality when compared to the original SDM system (Figure 5).

The Hamming Distance calculations used for all the experiments set the distance between a 0, or a 1 and a don't care (*) to 0.5. Although the tests indicate that this modification was highly beneficial, there may be scope for further improvement. When the distance between a 0, or 1 and a * is set to 0, patterns with don't care's act as extreme attractors. Additionally, if the distance between a 0, or 1 and a * is set to 1, the same patterns would act as repellers. Simulations that vary this distance by incrementally selecting values between 0 and 1 would be quite insightful.

Another worthy area of investigation is domain based initialization techniques. As the simulated tests revealed, even when the memory was filled to its capacity, about two thirds of the virtual processing cells (hard locations) were unused. Given the memory and computational constraints of the computing systems that software agents 'reside' in and hence, the constraints on the number of hard locations that such agents' TEM can have, we hypothesize that domain-based initialization mechanisms for the Modified SDM will further improve performance.

Conclusions

This paper argues that the modified SDM system is a suitable computational model for TEM in cognitive software agents. The simulations show that it outperforms the original SDM system in its ability to encode text based episodic data. Although, a simple text based domain was used in the work reported here, we speculate that the modified SDM system will show a significant performance improvement in a domain using less formal representations such as perceptual symbols (Barsalou, 1999). This motivates its use as a TEM for autonomous robots that are structurally coupled with the real world.

Acknowledgments

The second author is supported in part by NIH Cancer Center Support CORE grant, P30 CA-21765 and by the American Lebanese Syrian Associated Charities (ALSAC). The authors acknowledge the support of Dr. Lee McCauley, Matthew Ventura, Amy Witherspoon, and the Conscious Software Research Group (http://csrg.cs.memphis.edu).

References

Anwar, A., & Franklin, S. (2003). Sparse Distributed Memory for "Conscious" Software Agents. *Cognitive Systems Research, 4*, 339-354.

Baars, Bernard J. (1988). *A Cognitive Theory of Consciousness*. Cambridge: Cambridge University Press.

Baars, Bernard J. (1997). *In the Theater of Consciousness*. Oxford: Oxford University Press.

Baars, B. J., and S. Franklin. 2003. How conscious experience and working memory interact. *Trends in Cognitive Science* 7:166-172.

Baddeley, A. D. (2000). The episodic buffer: A new component of working memory? *Trends in Cognitive Science, 4*, 417-423.

Baddeley, A., M. Conway, & Aggleton, J. (2001). *Episodic Memory*. Oxford: Oxford University Press.

Conway, M. A. (2001). Sensory-perceptual episodic memory and its context: Autobiographical memory. In A. Baddeley, M. Conway, & J. Aggleton (Eds.), *Episodic Memory*. Oxford: Oxford University Press.

Fillmore, C. (1968). The case for case. In E. Bach & R. T. Harms (Eds.), *Universals in Linguistic Theory*,. New York: Holt, Rinehart and Wilson.

Franklin, S. (1997). Autonomous Agents as Embodied AI. *Cybernetics and Systems' Special issue on Epistemological Aspects of Embodied AI*, *28:6*, 499-520.

Franklin, S. (2001). Conscious Software: A Computational View of Mind. In V. Loia & S. Sessa (Eds.), *Soft Computing Agents: New Trends for Designing Autonomous Systems*. Berlin: Springer (Physica: Verlag).

Franklin, S., B. J. Baars, U. Ramamurthy, and M. Ventura. in review. The Role of Consciousness in Memory. .

Hely, T. A., Willshaw, D. J. & Hayes, G. M. (1997). A New Approach to Kanerva's Sparse Distributed Memory. *IEEE Transactions on Neural Networks*, *8(3)*, 791-794.

Kanerva, P. (1988). *Sparse Distributed Memory*. Cambridge MA: The MIT Press.

Kanerva, P. (1993). Sparse Distributed Memory and related models. In M. H. Hassoun (Ed.), *Associative Neural Memories: Theory and Implementation*. (pp. 50-76). New York: Oxford University Press

Keeler, J. D. (1988). Comparison Between Kanerva's SDM and Hopfield-Type Neural Networks. *Cognitive Science, 12:3*, 299-329.

Knuth, D. E., Morris, J. H. & Pratt, V. R. (1977). Fast pattern matching in strings. *SIAM Journal on Computing*, *6*, 323-350.

Ramamurthy, U., D'Mello, S. K., & Franklin, S. (May, 2003). Modeling Memory Systems with Global Workspace Theory. *Seventh Conference of the Association for the Scientific Study of Consciousness - ASSC7*

Ramamurthy, U., D'Mello, S., & Franklin, S. (October, 2004). Modified Sparse Distributed Memory as Transient Episodic Memory for Cognitive Software *Agents. Proceedings of the International Conference on Systems, Man and Cybernetics*. The Hague, Netherlands.

Shastri, L. (2002). Episodic memory and cortico hippocampal interactions *TRENDS in Cognitive Sciences*, *6:4*, 162-168.