International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:10, 2007

Review and Experiments on SDMSCue

Ashraf Anwar

***Abstract*—**In this work, I present a review on Sparse Distributed Memory for Small Cues (SDMSCue), a variant of Sparse Distributed Memory (SDM) that is capable of handling small cues. I then conduct and show some cognitive experiments on SDMSCue to test its cognitive soundness compared to SDM. Small cues refer to input cues that are presented to memory for reading associations; but have many missing parts or fields from them. The original SDM failed to handle such a problem. SDMSCue handles and overcomes this pitfall. The main idea in SDMSCue; is the repeated projection of the semantic space on smaller subspaces; that are selected based on the input cue length and pattern. This process allows for Read/Write operations using an input cue that is missing a large portion. SDMSCue is augmented with the use of genetic algorithms for memory allocation and initialization. I claim that SDM functionality is a subset of SDMSCue functionality.

***Keywords*—**Artificial intelligence, recall, recognition, SDM, SDMSCue.

I. Introduction

SPARSE Distributed Memory (SDM) is a content addressable memory developed by Kanerva [16]. SDM was proposed to be a tool and model of human associative memory [8, 13, 15, 16, 17]. SDM is a content-addressable memory technique that relies on similar memory items tending to be clustered together in the same region or subspace of the semantic space. SDM has been used before as associative memory or control structure for software agents.

SDM has proven successful in modeling associative memories [3, 20, 21, 22]. Associative memory is typically needed for intelligent and cognitive autonomous agents [12, 20]. In particular, both cognitive software agents [9] and "conscious" software agents [10] need such a memory. One of the "conscious" software agents, which I have worked with, IDA (Intelligent Distribution Agent), uses SDMSCue. I use SDMSCue in IDA to learn and keep associations between various pieces of information pertaining to the task of personnel distribution [11]. Auto-associative SDMSCue is indeed capable of recovering and recalling information using an arbitrary small part of that information; when the original SDM would fail. In general, we need to use SDMSCue whenever we want to overcome the pitfall in original SDM; of failing to handle small cues [4]. I conducted some cognitive experiments using SDMSCue and SDM; pertaining to recall, and recognition. The test results reveal superiority of SDMSCue over SDM in many aspects. I claim that except for chance convergence in original SDM; functionality of SDM is a subset of that of SDMSCue.

II. The Motive for SDMSCue

In many cases the need for associative memory to be able to handle and retrieve associations based on arbitrary small cues is crucial. For example; in IDA [11], we are often faced with a situation in which we need to retrieve associations based on very small pieces of information like part of email address, part of name, or social security number. Humans have no problem retrieving associations based on arbitrary small cues. While original SDM modeled many aspects of human memory very successfully [16], it failed miserably in dealing with the issue of retrieving associations based on short-length or small cues. Without such a capability, we are missing a key human-like feature [4, 7]; in associative memory models that are based on SDM. Hence, the role of SDMSCue comes to the scene.

SDMSCue uses an elegant space projection mechanism to *enlarge* the short-length input cue successively until it is large enough for a Read/Write from/to the entire full-length SDM semantic space [4]. The enlargement process uses successively increasing subspaces for reads/writes. To be noted is that both read and write operations in SDM involve the selection of an access circle to read from, or to write to. The selection is typically based on similarity between the input Read/Write cue, and the hard locations addresses within the access circle.

III. SDM for Small Cues

Here, I present **SDM** for **S**mall **Cue**s, SDMSCue, and contrast it when applicable to the original SDM.

A. Approach

Using a variant of SDM capable of handling small cues, we are able to overcome the main shortage in Kanerva's model [16]. One of the main problems with Kanerva's SDM is that the input cue has to be of sufficient length to be able to retrieve a match. The reason is that the *entire* input cue is considered and the hamming distance between its *entire* binary string representation and various hard locations in the access sphere or access circle is considered when reading or writing. So if we have a small cue, the missing large part is almost guaranteed to sink the known small cue in terms of hamming distance, thus being indifferent to all words or hard locations in the SDM memory.

In many cases, we are faced with a very distinguished and unique memory cue that is considerably small in size than the typical 65-80% requirement in Kanerva's work. We -humans- are able to retrieve relevant information associated with such a small cue efficiently. For SDM to be able to function similarly, we need a variant of SDM that is capable of handling small cues. When faced with retrieval based on a substantially short cue like part of name, part of email address, or social security number, this calls for the use of SDMSCue.

The goal is to retrieve appropriate corresponding word matching such a small cue. Using subspaces with increasing sizes in a progressive way, we are able to read and retrieve the whole original corresponding memory item using only a small

Ashraf Anwar is with the Gulf University for Science and Technology, Hawalli, 32093 Kuwait (e-mail: anwar.a@gust.edu.kw).

portion of the cue with arbitrary small length as a start. However, the number of levels needed for Read/Write depends upon the original size of the small cue. The smaller the original input cue is; the more levels of Read/Write needed. Time complexity of the operation is linearly proportional to the number of levels needed. Thus, the genuine gain in performance is quite superior to the added time complexity.

B. Design of SDMSCue

The idea is to project the original SDM semantic space onto a smaller subspace corresponding to the small cue. In such a projection, only memory locations with ***matching*** content to the small input cue contribute to Read/Write operations.

By projecting the space onto a smaller subspace, we are able to use the smaller subspace for much higher recall rate for a considerably small cue. The gain occurs mainly because of constraining and limiting only hard locations that match the small cue part and allowing only them to contribute to the subspace for Read/Write operations.

The result obtained from a Read/Write operation at one stage; is used to access a larger subspace including the input cue along with associations retrieved that typically range from 25% to 35% of the former input cue length. Such associations are retrieved from the contents of the hard locations that were selected, and contributed to the former subspace Read/Write.

By repeating the above process for increasingly larger subspaces and levels of projection, we eventually get to access the entire semantic space for Read/Write.

To be noted is that during write operations, actual writing to hard locations occurs only at the final level when writing to the entire space. All preceding access takes place for association retrieval only. So both read and write operations are the same (reading and retrieving associations) until the final level when we access the entire space. In both cases association buildup takes place to enlarge the small input cue gradually until the length obtained is large enough to read from the entire semantic space. In the last level or phase, if it is a read operation, we simply retrieve associations and obtain the matching entire word. If it is a write operation, the enlarged input cue is written to all hard locations within the access circle of the last phase, which is part of the entire semantic space.

C. How SDMSCue Works

Using SDMSCue, we can manage to access (Read/Write) with small cues. The process goes in phases in reading or writing operations. When accessing, in the first phase, we read from a small sub-space that corresponds to the input small cue plus extra association information. This read –if convergent- yields a longer word due to the association of information. This resulting word is then used as the input to the second phase. In the second phase, a similar process takes place reading from a larger subspace using the output result from the first phase as input. This process continues until the subspace being read from is the entire original semantic space.

For example, as shown in Table I, we start by reading with a small cue of length 17% of the whole memory word size, using a 0.35 ratio for associations. Then the reading operation yields a larger word, due to adding associations, of length 23% of the whole memory word size. Then using the 23% retrieved and formed word as input to the second phase and adding 0.35 associations to it, a 31% word is obtained. The process continues until in the final level (6^{th} level in the example), a 77% retrieved and formed word is used to access (read from or write to) the entire original semantic space.

To be noted is that the time complexity of a Read/Write operation is ***linearly*** proportional to the number of levels involved in a Read/Write. However, the overall effect is quite minor compared to the gain of the approach.

Assume an original cue length of ***m***% of the whole memory word size; where **m** ranges from 0 to 100. When reading/writing from SDMSCue, some associations are retrieved for the small cue at each level resulting in a length gain.

TABLE I
MULTI-LEVEL READING OPERATIONS FROM SDMSCue

Level #	Input Word Length	Output Word Length	Output Word
1	17	23	|-----|
2	23	31	|--------|
3	31	42	|----------|
4	42	57	|----------|
5	57	77	|----------|
6	77	100	|----------|

Let ***i*** be the percentage of the length gain at each level. Adding the gain in length, ***i,*** to the next input cue in each successive Read/Write level, the maximum number of Read/Write levels ***N;*** is given by:

$100 \le \boldsymbol{m} * (1 + \boldsymbol{i})^{N}$; Last word length needs to be 100%, i.e. last read needs to occur from the whole space

$$\Rightarrow N = \lceil \log(100/m) / \log(1+i) \rceil$$
$$\Rightarrow N = \lceil (2 - \log m) / \log(1+i) \rceil$$

For ***m*** = 17 (17% original small cue length), ***i*** = 0.35 (35%), as in *Table I*,

$$N = \lceil (2 - \log 17) / \log 1.35 \rceil = \lceil 5.9 \rceil = 6 \text{ Levels.}$$

For ***m*** = 10 (10% original small cue length), ***i*** = 0.3 (30%),

$$N = \lceil (2 - \log 10) / \log 1.3 \rceil = \lceil 8.78 \rceil = 9 \text{ Levels.}$$

For ***m*** = 1 (1% original small cue length), ***i*** = 0.3 (30%),

$$N = \lceil (2 - \log 1) / \log 1.3 \rceil = \lceil 17.55 \rceil = 18 \text{ Levels.}$$

SDMSCue Latency Factor [4] is the average number of levels needed for Read/Write for a certain word distribution to be written to or read from SDMSCue. Such a factor is both semantic space dependent, and distribution dependent.

SDMSCue makes use of GA for more efficient space initialization and hard locations allocation [1, 2]. The uniformity of the semantic space is –in general- favorable to better recall rates for SDMSCue as well as SDM.

D. SDMSCue Convergence and Divergence

We need to develop a notion for overall convergence and divergence in case of Read/Write from SDMSCue. For overall convergence to occur, all phases or levels of Read/Write must converge. Overall divergence occurs if at any phase or level, the Read/Write operation diverges. In other words, convergence in SDMSCue is the Boolean "AND" operation of the convergence in all levels.

$$\text{Convergence}|_{\text{SDMSCue}} = \text{AND}_{\text{For All } i}(\text{convergence}|_{\text{level } i})$$
$$\text{Divergence}\ |_{\text{SDMSCue}} = \text{NOT}\ (\text{Convergence}|_{\text{SDMSCue}})$$

The issue of false-positive (SDMSCue recalls some word based on an input cue that was never stored before) might be of concern here in case of memory convergence. Two cases need to be considered. Firstly, when the original input cue is of sufficient length, SDMSCue reverts in functionality to SDM reading with one level only. Secondly, when the input cue is small, it requires multiple reading levels. In this case, on average, this should not increase the rate of false-positive since the final reading occurs using a sufficiently long cue from the full memory (similar to SDM). The input for the reading operation at the last level should project to the same original small input cue with a high probability if number of reading levels is limited. However, in some extreme cases, false-positive rate might increase compared to SDM if each successive level of reading in SDMSCue presents noise (toggles one or more bits) to the bits of the very original small input cue. On the other hand, we should consider the huge gain achieved by employing SDMSCue vs. SDM, which greatly outweighs the issue of false-positive. On the other hand, false-negative is not really a concern. This is because SDM typically diverges, except for chance convergence, when presented with a small cue.

E. Implementation

The following is a short note about the current implementation of SDM with small cues (SDMSCue). Java Visual Symantec Café Professional Edition was used for testing and implementation of the code for SDMSCue in Windows XP environment. The hardware was a Pentium 2.4 GHz with 1GB RAM. The results obtained are based on recall performance and memory trace used for comparison tests of SDMSCue vs. original SDM. Runs were performed repetitively 100 times on average for each case. Other implementations and implementation platforms may be considered in future research.

IV. Experiments

The experiments conducted here relate to SDMSCue vs. SDM cognitive capabilities [4, 6]. Mainly recall and recognition are considered here. More tests on other cognitive capabilities including, but not limited to, deliberation and reasoning will be target for future research..

A. Recall

The following comparison between SDMSCue and regular SDM was done using the same memory parameters, and memory trace [19]. Memory performance in terms of various operational parameters was considered for SDMSCue vs. original SDM. The various memory parameters: Memory Volume, Cue Volume, Similarity, and Noise were considered.

Memory Volume is the average number of features in the memory trace. In other words, it is the average number of 1's in a memory word. It measures the richness of the memory trace. Memory volume is a vital parameter in the distinction of the memory trace. It signifies the distribution of various memory words over the semantic space.

Retrieval Volume is the same as Memory volume but for a single input cue or input word to memory. It has almost the same effect on retrieval as memory volume.

Similarity is a measure of how similar, in average, are the words written to memory. The more similar the words written to memory are, the more clustered contiguously they are, and the harder it is to retrieve them. The hamming distance is the measure of similarity in SDM as well as SDMSCue. The less the hamming distance between two memory words, the more similar the memory words are. However, there is a difference between the similarity of hard locations and the similarity of written memory words. Using genetic algorithms [1], a uniform distribution of the hard locations in SDM can be obtained.

Noise determines the number of noise bits, on average, in a memory word. It reflects directly on the reliability of retrieval of stored memory words.

Table II shows the distribution of the percentage of input cues in memory trace, used for the test, with respect to cue length, measured as percentage of the whole length. For example, according to Table II, 35% of the cues in memory trace do not exceed 20% in length (small cues), while 25% of the cues in memory trace have length greater than 20% but less than or equal to 40% (low medium cues). Also only 10% of the input cues have length greater than 70% (longest cues). The second column gives the number of levels needed for Read/Write operation using the formula devised in section 4.3.

TABLE II
DISTRIBUTION OF INPUT CUES IN MEMORY TRACE WITH RESPECT TO CUE LENGTH

Cue Minimum-Maximum Length as % of the Whole Word	Average # of Read/Write Levels Needed	Percentage in Memory Trace
Less than 20%	8	35%
20%-40%	5	25%
40%-50%	3	10%
50%-60%	2	10%
60%-70%	2	10%
70%-100%	1	10%
Average Length = 36%	Latency Factor = **5**	

As shown in Table II, for the chosen distribution, the overall average cue length is 36%, and the average number of levels for Read/Write is 5. So, for the distribution at hand, SDMSCue has a latency factor of 5.

To be noted is that this distribution was chosen to illustrate the advantage of using SDMSCue when considerable percentage of the input cues is short in length, i.e. missing too many parts. By no means is this the only distribution that can illustrate the idea, but just the one we settled upon after some trials to illustrate the benefit of using SDMSCue when considerable number of the input cues is short in length.

However, varying the distribution will definitely change the gain achieved from using SDMSCue over SDM.

Table III shows a comparison between the recall in SDM vs. SDMSCue. Various combinations of the memory trace parameters were considered. Each was varied on a Low/High scale.

The gain achieved from using SDMSCue is illustrated in the last two columns. The first gain, Recall Gain, measures the improvement in successful recall or Hit in SDMSCue over original SDM.

The second gain, Decrease in Miss in Recall, measures the decrease in miss rate in SDMSCue over the original SDM. In other words, it measures the improvement in SDMSCue over SDM in terms of decrease in the percentage of unsuccessful recall, or memory miss rate.

To be noted is that a T-Test of statistical dependence [18, 23] shows statistical significance between the recall of SDMSCue and that of SDM.

In Table III, Row 4, which represents poor recall conditions (Low Memory Volume, Low Retrieval Volume, High Similarity, and High Noise), shows large improvement in successful recall in SDMSCue over original SDM.

Row 13, on the other hand, represents near optimal recall conditions (High Memory Volume, High Retrieval Volume, Low Similarity, and Low Noise). Row 13 corresponds to 36% recall gain.

TABLE III
COMPARISON BETWEEN GA-INITIALIZED SDM AND GA-INITIALIZED SDMSCue. MEMORY VOLUME AND RETRIEVAL VOLUME (H=60%, L=10%). SIMILARITY BETWEEN MEMORY ITEMS (H=70%, L=30%). NOISE (H=30%, L=10%).

#	Memory Volume	Retrieval Volume	Similarity	Noise	SDM Hit % in Recall	SDMSCue Hit % in Recall	Recall Gain %	Decrease in Miss in Recall Gain %
1	L	L	L	L	6	44	633	40
2	L	L	L	H	5	40	700	37
3	L	L	H	L	7	39	457	34
4	L	L	H	H	5	34	580	31
5	L	H	L	L	9	56	522	52
6	L	H	L	H	8	50	525	46
7	L	H	H	L	11	49	345	43
8	L	H	H	H	5	42	740	39
9	H	L	L	L	74	98	32	92
10	H	L	L	H	66	93	41	79
11	H	L	H	L	61	93	52	82
12	H	L	H	H	54	89	65	76
13	H	H	L	L	73	99	36	96
14	H	H	L	H	56	95	70	89
15	H	H	H	L	60	96	60	90
16	H	H	H	H	50	92	84	84

For decrease in miss in recall gain, recall conditions play similar role. Row 4, which represents poor recall conditions, shows 31% gain. Row 13, which represents near optimal recall conditions, shows 96% gain.

In general, results in Table III show that the higher the volume of the memory and/or the retrieval volume, the better the recall for both memories, SDMSCue and SDM. However, when the memory volume is quite low, the recall gets really affected. The degradation in performance is more grace in SDMSCue than it is in SDM. This has to do with the fact that SDMSCue projects the space on the part of the cue that exists, thus greatly moderating the typical negative effect of low memory volume.

With respect to similarity, the more distinct the memory words are, i.e. the less similarity, the better the recall in general. This makes absolute sense since SDM in general; and accordingly SDMSCue as well, uses hamming distance as a measure of inclusion of memory words in access sphere or access circle for Read/Write. The more similar memory words are, the closer they become in terms of hamming distance. Hence, they tend to get clustered together and cause crowding effect in the semantic space, where they may sink each other in certain regions of the semantic space.

To be noted, though, is that such clustering effect in the semantic space resulting from highly similar items being written to SDM or SDMSCue is somewhat similar but not the same as clustering of hard locations when a poor initialization technique is used for hard locations assignment. The later is more crucial to the functioning of SDM or SDMSCue, and should be application independent. Whereas the former depends on the memory trace fed to SDM or SDMSCue, and hence is application dependent by nature. While there is a straightforward way to guarantee uniformity of hard locations assignment in the semantic space of SDM or SDMSCue [2], there is no immediate direct solution to overcome clustering in SDM or SDMSCue due to similarity of words or memory items written to it.

As expected, for the effect of noise on recall; the lower the noise, the better the recall in general. Noise can however be sunk to a degree as a result of the distributed nature of Read/Write, as well as the abstraction achieved from using SDM and SDMSCue.

Fig. 1 contrasts the performance of SDMSCue vs. SDM in terms of hit rate in recall. The top line shows the recall hit rate in SDMSCue. The bottom line shows the recall hit rate in SDM. The X-axis is for the 16 different memory parameters configurations from Table III. The Y-axis is for the percentage of Recall Hit Rate.

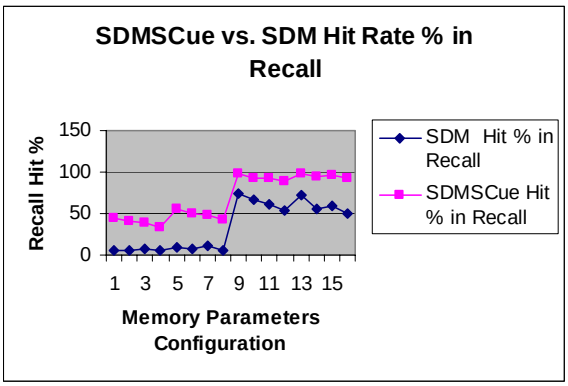

Fig. 1 SDMSCue vs. SDM hit rate % in recall

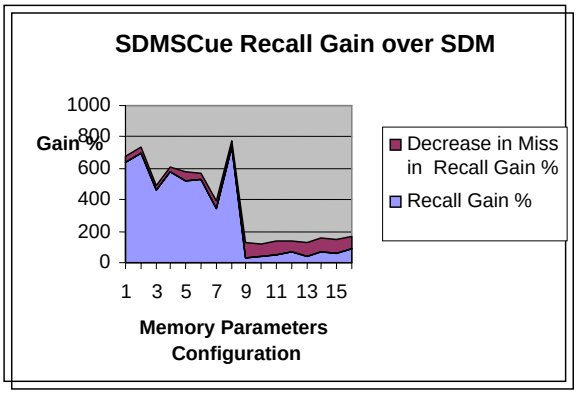

Fig. 2 SDMSCue recall gain over SDM

In this comparison, the 16 different memory configurations in Table III were used, e.g. configuration 9 is (HLLL) which stands for High Memory Volume, Low Retrieval Volume, Low Similarity, and Low Noise. For each configuration, the same memory trace was applied to both SDM and SDMSCue. Then recall was tested with the same set of patterns with lengths distributed according to Table II. Recall hit rate for each case was calculated and is shown as percentage over the vertical axis in Fig. 1.

It is clear from Fig. 1 that SDMSCue consistently outperforms SDM in terms of recall under all conditions. This comes at no surprise since SDMSCue uses SDM functionality in addition to the elegant space-projection to filter out non-relevant memory locations. SDMSCue also uses a far more superior GA approach for uniform space initialization and allocation of hard locations [1].

Fig. 2 shows the gain of SDMSCue over original SDM using the results and figures from Fig. 1. The results were obtained using Fig. 1. The bottom area shows the gain in recall. The upper area shows the improvement in recall measured as the decrease in miss rate in recall.

The bottom area is the recall gain in SDMSCue over SDM. The gain percentage of SDMSCue over SDM is defined by: 100*(SDMSCue Hit% – SDM Hit%) / (SDM Hit%). For example, for memory parameter configuration 1, the gain is 100 * (44 - 6) / 6 = 633%.

The upper area is the improvement in recall measured as the decrease in miss rate in recall. This is defined by: 100*(SDM Miss% – SDMSCue Miss%) / (SDM Miss%). For example, for memory parameter configuration 1, the improvement as decrease in miss rate is 100*(94 – 56) / 94 = 40%. For memory parameter configuration 13, the improvement as decrease in miss rate is 100*(27 – 1) / 27 = 96%.

B. Recognition

For recognition [7, 19], the "meaning" of information or data is recognized; rather than the mere information or data.

Per-se, when SDMSCue enhances recall as shown in 5.1, recognition has a room for improvement. Firstly, the retrieval or recall based on arbitrary small cue is one way to achieve simple recognition. In this retrieval, a meaning could be stored in SDMSCue with a key or cue for access and retrieval. In other words, when SDMSCue goes into a reading cycle expanding the input cue through level-reading; this is arguably a form of this simple recognition. Secondly, more sophisticated recognition can be achieved more easily with SDMSCue vs. SDM. Any form of meaning association, conceptualization, or abstraction can make use of the small cue extra functionality provided by SDMSCue to some extent.

An accurate measure of recognition improvement is typically domain dependent. However, a crude general measure of recognition and/or recognition improvement may be devised for our test purposes as follows:

a. for SDM, build two sparse distributed memories: auto-associative and hetero-associative
b. in the first, the auto-associative memory, store concepts in a certain domain. For example, in computer literacy domain, what is external storage, what is data communication, etc…
c. in the second, the hetero-associative memory, store elaborations or meanings of the concepts stored in b.
d. the output from b is fed as input to c. This way, the concept retrieved is elaborated upon or meanings for it are extracted.
e. for SDMSCue, build two corresponding memories: one auto-associative, and one hetero-associative and store same data. Same operations as in b, c, and d.
f. start retrieval from both SDM and SDMSCue with an input cue.
g. examine the chain of associations retrieved from the hetero-associative memory for SDM vs. the one for SDMSCue. This chain or sequence may represent the thought sequence provoked by an idea or a concept that flashes in mind. Deliberation, reasoning, and introspection [5, 14] on this sequence is yet another task that can be performed.
h. The hamming distance between the input cue and the mean word of the output convergent sequence from

hetero SDMSCue vs. hetero SDM is considered. The lesser, the better.

i. The sum of the hamming distances between the input cue and the first ***K*** words in the output sequence, divided by ***K*** (*K*=10 for results shown). Hetero SDMSCue vs. hetero SDM are considered and shown. The lesser, the better.

When comparing recognition between SDMSCue and original SDM according to the method described above, a system like the one shown in Fig. 3 may be used.

A sample run of the system using the same data set used in recall experiments along with words for the hetero-associative version yields the results shown in Fig. 4, and Fig. 5. Top line is for hetero SDM, and bottom line is for hetero SDMSCue. The figures show hamming distance as a function of input cue word length for SDMSCue vs. SDM. Fig. 4 is for the hamming distance between the input cue and mean word of the output convergent sequence (as described in h) on the Y-axis, vs. the input cue length on the X-axis.

As the graph shows, SDMSCue significantly has a lesser hamming distance between the input cue word and the mean word of the hetero SDMSCue output sequence. This is true for all values of the x-axis which represents the lengths of the input cue words.

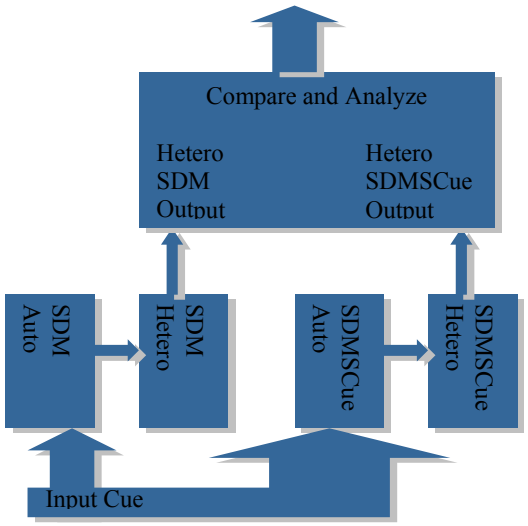

Fig. 3 The layout for the comparison of recognition of SDMSCue vs. SDM

On the other hand, SDM does not even approach the performance of SDMSCue until the input cue length is of sufficient length as shown in Fig. 4. In such a case, SDM converges to match SDMSCue performance as shown. The huge gap is in part due to the big word size (10,000 bits) and the starting input cue length of 10 bits. As might be expected, if the start length of the input cue is large, the gap will not be as huge as shown in Fig. 4.

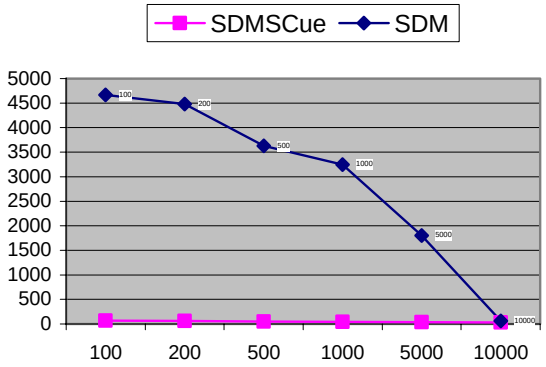

Fig. 4 Hamming distance for mean word as a function of input cue word length

Fig. 5 is for the average hamming distance (sum of hamming distances between the input cue and the first ***K*** words in the output sequence, divided by *K*); as described in i, on the Y-axis vs. the input cue length on the X-axis. Again, the performance of SDMSCue vs. SDM is contrasted. Due to similar arguments, SDMSCue again outperforms SDM for all small cue lengths. SDM only aspires to catch up with SDMSCue for sufficiently large input cue words. Again the gap is huge due to the same arguments given earlier.

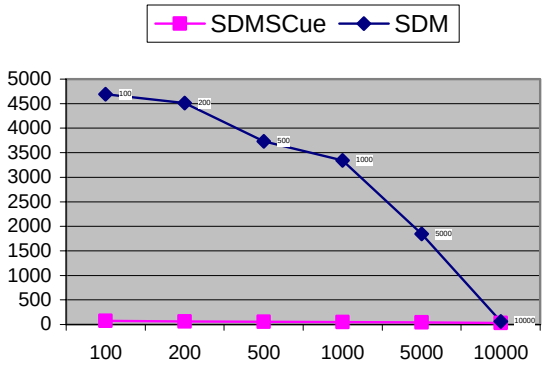

Fig. 5 Average hamming distance as a function of input cue word length

V. Conclusion

SDMSCue is superior to traditional SDM in its capability of handling small cues that original SDM was not able to. One of the major difficulties encountered in using original SDM as an associative memory, is its inability to recover associations based upon relatively small cues; whereas we humans do. For a typical SDM to converge, a sufficiently large portion of a previously written word must be presented to the memory as an address. The SDMSCue enhanced version of SDM, allows for handling small input cues and overcoming these pitfalls. Such cues were beyond the scope of original SDM work. The ability of SDMSCue to overcome the input cue length constraint in the original SDM model provides superior functionality for associative memory. It allows for association and matching based on small hints or input cues. The recall

results obtained for SDMSCue are –in general- superior to those of original SDM. The gain achieved is quite significant statistically as well as objectively. SDMSCue recognition as defined here; is also quite superior to that of original SDM.

VI. Future Research

More comparisons and tests of SDMSCue vs. original SDM in cognitive domains are underway. Deliberation and reasoning using both techniques is being considered. Applying SDMSCue as an associative memory technique for some existing intelligent architectures; is also under consideration. A more thorough analysis of the relation between number of reading levels and input data distribution vs. false-positive rate may also be done.

References

[1] Anwar, Ashraf, 'Sparse Distributed Memory with Evolutionary Mechanisms', Proceedings of Genetic and Evolutionary Computation Conference Workshop (GECCO), 1999, pp. 339-40.
[2] Anwar, Ashraf, Dasgupta, Dipankar, and Franklin, Stan, Using Genetic Algorithms for Sparse Distributed Memory Initialization, Proceedings of Congress on Evolutionary Computation (CEC), 1999.
[3] Anwar, Ashraf, and Franklin, Stan, Sparse Distributed Memory for "Conscious" Software Agents, Cognitive Systems Research Journal, UK: Elsevier, December 2003, v 4 n 4, pp. 339-54.
[4] Anwar, Ashraf, and Franklin, Stan, A Sparse Distributed Memory Capable of Handling Small Cues, SDMSCue, IFIP International Federation for Information Processing, USA: Springer Science+Business Media Inc., Formerly Kluwer Boston Inc, 2005, vol. 172, pp. 23, ISSN: 1571-5736 (Paper) 1861-2288 (Online).
[5] Doyle, John, A Model for Deliberation, Action, and Introspection (AI TR), USA: MIT, AI Lab, 1980.
[6] Evans, Richard, and Surkan, Alvin, Relating Number of Processing Elements in a Sparse Distributed Memory Model to Learning Rate and Generalization, APL Quote Quad, Aug 1991, v 21, n 4, pp. 166.
[7] Feldman, Robert S., Understanding Psychology, USA: McGraw Hill, 2005.
[8] Franklin, Stan, Artificial Minds, USA: MIT Press, 1995.
[9] Franklin, Stan, Autonomous Agents as Embodied AI, Cybernetics and Systems Journal, special issue on Epistemological Issues in Embedded AI, 1997.
[10] Franklin, Stan, and Graesser, Art, A Software Agent Model of Consciousness, Consciousness and Cognition Journal, 1999, v 8, pp. 285-305.
[11] Franklin, Stan, Kelemen, Arpad, and McCauley, Lee, IDA: A Cognitive Agent Architecture, IEEE Transactions on Systems, Man, and Cybernetics, USA: IEEE, NJ, 1998.
[12] Glenberg, Arthur M., What Memory is for?, Behavioral and Brain Sciences Journal, USA: Cambridge University Press, 1997.
[13] Hely, T., The Sparse Distributed Memory: A Neurobiologically Plausible Memory Model?, Master's Thesis, UK: Edinburgh University, Department of Artificial Intelligence, 1994.
[14] Ingrand, F. F., and Georgeff, M. P., Managing Deliberation and Reasoning in Real-Time AI Systems, Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control, pp. 284-91, 1990.
[15] Kanerva, Pentti, and Raugh, Michael, Sparse Distributed Memory, RIACS, Annual Report, USA: NASA Ames Research Center, Moffett Field, CA, 1987.
[16] Kanerva, Pentti, Sparse Distributed Memory, USA: MIT Press, 1988.
[17] Kanerva, Pentti, The Organization of an Autonomous Learning System, USA: RIACS-TR-88, NASA Ames Research Center, Moffett Field, CA, 1988.
[18] Kanji, Gopal K., 100 Statistical Tests, USA: Sage Publications, 1999.
[19] Loftus, Geoffrey, and Loftus, Elizabeth, Human Memory, the Processing of Information, USA: Lawrence Erlbaum Associates, 1976.
[20] Rao, Rajesh P. N., and Fuentes, Olac, Learning Navigational Behaviors using a Predictive Sparse Distributed Memory, Proceedings of the 4th international Conference on Simulation of Adaptive Behavior, From Animals to Animats IV, 1996, pp. 382.
[21] Rao, Rajesh P. N., and Fuentes, Olac, Hierarchical Learning of Navigation Behaviors in an Autonomous Robot using a Predictive Sparse Distributed Memory, Machine Learning Journal, April 1998, v 31, n 1/3, pp. 87-113.
[22] Scott, E., Fuller, C., and O'Brien, W., Sparse Distributed Associative Memory for the Identification of Aerospace Acoustic Sources, AIAA Journal, September 1993, v 31, n 9, pp. 1583.
[23] Vogt, W. Paul, Dictionary of Statistics and Methodology, 2nd Edition, USA: Sage Publications, 1998.