Sparse Distributed Memory for “Conscious” Software Agents

A. Anwar
S. Franklin

Follow this and additional works at: https://digitalcommons.memphis.edu/ccrg_papers

Recommended Citation
Sparse Distributed Memory for “Conscious” Software Agents

Ashraf Anwar  
Institute for Intelligent Systems  
The University of Memphis  
Memphis, TN 38152  
aanwar1@memphis.edu

Stan Franklin  
Institute for Intelligent Systems  
The University of Memphis  
Memphis, TN 38152  
stan.franklin@memphis.edu

Abstract- In this work we are reporting a case study on the use of SDM as the associative memory for a software agent, CMattie, whose architecture is modeled on human cognition. Sparse Distributed Memory (SDM) is a content-addressable memory technique that relies on close memory items tending to be clustered together. In this work, we used an enhanced version of SDM augmented with the use of genetic algorithms as an associative memory in our “conscious” software agent, CMattie, who is responsible for emailing seminar announcements in an academic department. Interacting with seminar organizers via email in natural language, CMattie can replace the secretary who normally handles such announcements. SDM is a key ingredient in a complex agent architecture that implements global workspace theory, a psychological theory of consciousness and cognition. In this architecture, SDM, as the primary memory for the agent, provides associations with incoming percepts. These include disambiguation of the percept by removing noise, correcting misspellings, and adding missing pieces of information. It also retrieves behaviors and emotions associated with the percept. These associations are based on previous similar percepts, and their consequences, that have been recorded earlier. SDM also possesses several key psychological features. Some enhancements to SDM including multiple writes of important items, use of error detection and correction, and the use of hashing to map the original information into fixed size keys were used. Test results indicate that SDM can be used successfully as an associative memory in such complex agent architectures. The results show that SDM is capable of recovering a percept based on a part of that percept, and finding defaults for empty perception registers. The evaluation of suggested actions and emotional states is satisfactory. We think that this work opens the door to more scientific and empirical uses for SDM.

Keywords  
Artificial Intelligence, Cognition, Consciousness, Genetic Algorithms, Software Agents, Sparse Distributed Memory.

1 Introduction

Sparse Distributed Memory (SDM) is a content addressable memory developed by Kanerva (1988a), which we think is a useful model of human associative memory.

SDM has previously proved successful in implemented associative memories (Anwar, 1997; Hely, 1994; Scott, Fuller, & O’Brien, 1993). Associative memory is typically needed for intelligent autonomous agents (Glenberg, 1997; Kosslyn, 1992). In particular, both cognitive software agents (Franklin, 1997) and “conscious” software agents (Franklin & Graesser, 1997) need such a memory. CMattie, our clerical software agent who distributes weekly departmental seminar announcements (Bogner, Ramamurthy, & Franklin, 2000; Macauley & Franklin 1998; Ramamurthy, Bogner, & Franklin, 1998; Zhang, Franklin, & Dasgupta, 1998a; Zhang, Franklin, Olde, Wan, & Graesser, 1998b) uses SDM for associative memory. IDA, Intelligent Distribution Agent, also needs such a memory to learn and keep associations between various pieces of information pertaining to the task of personnel distribution (Franklin, Kelemen, & Macauley, 1998).

CMattie, our clerical software agent, uses an extended version of SDM (see section 6.2) to build, store, and retrieve associations between various percepts, behaviors, and emotions. Percepts are dissected into perception registers (PRs). Each perception register contains some key piece of information pertaining to a seminar, such as seminar organizer, speaker, date, location, etc. Behaviors are handled via an extended form of Maes’ behavior net (Maes, 1990). Emotions are implemented by a mechanism based on pandemonium theory (Jackson, 1987; Franklin, 1995; Macauley & Franklin, 1998). CMattie’s “consciousness” mechanism follows global workspace theory (Baars, 1988; 1997). SDM in CMattie is enhanced in many ways over the original Kanerva model (Anwar, Dasgupta, & Franklin, 1999).

Thus, CMattie in effect combines quite a variety of intelligent mechanisms aspiring to achieve human-like intelligence. It follows the theory that human intelligence is made out of disparate mechanisms. CMattie is a first step toward more complex and human-like agents. IDA, Intelligent Distribution Agent, is one such (Franklin, 2001; Kondadadi & Franklin, 2001).
2 Autonomous, Cognitive, and “Conscious” Agents

We adopt a general definition of autonomous agents (Franklin & Graesser, 1997) that reflects a broad sense of the term and its possible uses. According to this definition, an autonomous agent is a system situated in an environment that senses it, acts upon it, over time, so that its actions may affect what it senses next, and its actions are in pursuit of its own agenda. This definition covers a whole spectrum of agents varying from the very simple ones, e.g. a thermostat, to the most complex ones, e.g. a human.

A cognitive agent, on the other hand, is an autonomous agent, which has more cognitive capabilities: problem solving, planning, learning, perception, emotions...etc (Franklin, 1997; Perlovsky, 1999; 2002). Some examples of cognitive agents are TOK (Bates, Loyall, & Reilly, 1991; 1992), Emotional Agent (Wright, 1994; 1996), and TLCA (Anwar, 1997).

A “conscious” agent is a cognitive agent with the extra functionality of “consciousness” built in (Franklin & Graesser, 1999). We adopt consciousness a la Baars’ global workspace theory (Baars, 1988; 1997). One key feature of consciousness is the broadcast of information in order to recruit processes that can help in handling the current novel or problematic situation.

Our definition of consciousness is mainly pertinent to satisfying the network structure according to Baars’ global workspace theory. Our agent may not have qualia-type experience, i.e. knowing what is it like to be a clerical agent. However, we believe in that it has some kind of self-awareness; proprioception, in that it is observing or performing a certain task; or proprio-consciousness, that it recognizes it is indeed conscious. Alternative consciousness theories do exist (Grossberg, 1999). We emphasize that we are just adopting Baars theory because of some of its architectural benefits to our research, and as an implementation decision. Other theories of consciousness could have been used as well.

Our agent, CMattie, is a “conscious” software agent, which implies that it is autonomous and cognitive as well. CMattie does indeed possess many human-like cognitive features.

3 Sparse Distributed Memory
3.1 A Brief Tour of SDM

Sparse Distributed Memory, SDM, is the work of Pentti Kanerva (1988a, 1988b). It gets its name from the sparse allocation of storage locations in a vast binary address space and from the distributed nature of information storage and retrieval, as we will explain later. A typical SDM has a vast binary space of possible memory locations in a $2^n$ semantic space where $n$ is both the word length and the dimension of the address space. For any practical application, only a very small portion of that space can actually exist. For more details and discussions, see section 3.2. Also see (Franklin, 1995) for a brief overview, (Willshaw, 1990) for a useful commentary, and (Anwar, 1997) for a more detailed summary of its mechanism. Many evaluations, extensions, and enhancements have been suggested for SDM (Evans & Surkan, 1991; Karlsson, 1995; Kristoferson, 1995a; 1995b; Rogers, 1988a, 1988b; Ryan & Andreae, 1995). A more efficient initialization technique for SDM using Genetic Algorithms was also found (Anwar et al, 1999).

There are two main types of associative memory, auto-associative and hetero-associative. In auto-associative memory, a memory item is used to retrieve itself. In hetero-associative memory, memory items are stored in sequences where one item leads to the next item in the sequence. The auto-associative version of SDM is truly an associative memory technique where the contents and the addresses belong to the same space and are used alternatively.

The word length, which is also the dimension of the space, determines how rich in features each word and the overall semantic space are. Features are represented by one or more bits in a Boolean vector or binary string of length $n$. Groups of features are concatenated to form a word, which becomes a candidate for writing into SDM. Another important factor is how many real memory locations are implemented. We call these hard locations. When writing, a copy of the object binary string is placed in all close enough hard locations. When reading, a cue would reach all close enough hard locations and get some sort of aggregate or average return from them. Reading is not always successful. Depending on the cue and the previously written information, among other factors, convergence or divergence during an iterative reading operation may occur. If convergence occurs, the pooled word will be the closest match of the input reading cue, possibly with some sort of abstraction. On the other hand, when divergence occurs, there is no relation, in general, between the input cue and what is retrieved from SDM.

SDM is a content addressable memory that, in many ways, is ideal for use as a long-term associative memory. “Content addressable” means that items in memory can be retrieved by using part of their contents as a cue, rather than having to know their actual addresses in memory as in traditional computer memory.

Boolean geometry deals with Boolean spaces. A Boolean space is the set of all Boolean vectors (points) of some fixed length, $n$, called the dimension of the space. The Boolean space of dimension $n$ contains $2^n$ Boolean vectors, each of length $n$. The number of points increases exponentially as the dimension increases. Boolean geometry uses a metric called Hamming Distance, where the distance between two points is the number of coordinates at which they differ. Thus $d((1,0,0,1,0), (1,0,1,1,1)) = 2$. The distance between two points will measure the similarity between two memory items, closer points being more similar. We may think of these Boolean vectors as feature vectors, where each feature can be only on, 1, or off, 0. Two such feature vectors are closer together if more of their features are the same.
For n = 1000, most of the address space lies between a distance of 422 and a distance of 578 from a given vector (Kanerva, 1988a). In other words, almost all the space is far away from any given vector. A Boolean space implementation is typically sparsely populated, an important property for the construction of the model, and the source of part of its name.

An SDM address space is a Boolean space. For n=1000, one cannot hope to actually implement such a vast memory. On the other hand, thinking of feature vectors, a thousand features wouldn’t deal with human visual input until a high level of abstraction had been reached. A dimension of 1000 may not be all that much; it may, for some purposes, be unrealistically small. Kanerva proposes to deal with this vast address space by choosing a uniform random sample, of size $2^{20}$, of hard locations, that is, about a million of them. An even better way to distribute the set of hard locations over the vast semantic space using Genetic Algorithms has been found (Anwar et al, 1999). With $2^{20}$ hard locations out of $2^{1000}$ possible locations, the ratio is $2^{-980}$, very sparse indeed.

SDM is distributed in that many hard locations participate in storing and retrieving each datum, and one hard location can be involved in the storage and retrieval of many data. This is a very different beast from the store-one-datum-in-one-location type of memory to which we are accustomed. For n = 1000, each hard location, itself a bit vector of length 1000, stores data in 1000 counters, each with range from -40 to 40. We now have about a million hard locations, each with a thousand counters, totaling a billion counters in all. Numbers in the range -40 to 40 will take most of a byte to store. Thus we are talking about a billion bytes, a gigabyte, of memory.

Updating counters depends on the data written. Writing a 1 to the counter increments it; writing a 0 decrements it. A datum, $\zeta$, to be written is a bit vector of length 1000. To write $\zeta$ at a given hard location $x$, write each coordinate (space dimension) of $\zeta$ to the corresponding counter in $x$, either incrementing it or decrementing it.

For n = 1000, call the sphere of radius 451 centered at location $\zeta$ the access sphere of that location. An access sphere typically contains about a thousand hard locations, with the closest to $\zeta$ usually some 424 bits away and the median distance from $\zeta$ to hard locations in its access sphere about 448. Any hard location in the access sphere of $\zeta$ is said to be accessible from $\zeta$. With this machinery in hand, we can now write distributively to any location, hard or not. To write a datum $\zeta$ to a location $\zeta$, simply write $\zeta$ to each of the roughly one thousand hard locations accessible from $\zeta$.

With our datum distributively stored, the next question is how to retrieve it. With this in mind, let us ask first how one reads from a single hard location, $x$. Compute $\zeta$, the bit vector read at $x$, by assigning its $i^{th}$ bit the value 1 or 0 according as the $i^{th}$ counter at $x$ is positive or negative. Thus, each bit of $\zeta$, results from a majority rule decision of all the data that have been written at $x$. The read datum, $\zeta$, is an archetype of the data that have been written to $x$, but may not be any one of them. From another point of view, $\zeta$ is the datum with smallest mean distance from all data that have been written to $x$.

Knowing how to read from a hard location allows us to read from any of the 21000 arbitrary locations. Suppose $\zeta$ is any location. The bit vector, $\zeta$, to be read at $\zeta$, is formed by pooling the data read from each hard location accessible from $\zeta$. Each bit of $\zeta$ results from a majority rule decision over the pooled data. Specifically, to get the $i^{th}$ bit of $\zeta$, add together the $i^{th}$ bits of the data read from hard locations accessible from $\zeta$, and use half the number of such hard locations as a threshold. At or over threshold, assign a 1. Below threshold assign a 0. Put another way, pool the bit vectors read from hard locations accessible from $\zeta$, and let each of their $i^{th}$ bits votes on the $i^{th}$ bit of $\zeta$.

To consider the relation between the datum in and the datum out, let's first look at the special case where the datum $\zeta$ is written at the location $\zeta$. This makes sense since both are bit vectors of length one thousand. Kanerva offers a mathematical proof showing that reading form $\zeta$ recovers $\zeta$ (Kanerva, 1988a). Here's the idea of the proof. Reading from $\zeta$ recovers archetypes from each of some thousand hard locations and takes a vote. The voting is influenced by about 1000 stored copies of $\zeta$ and, typically, by about 10,000 other stored data items. Since the intersection of two access spheres is typically quite small, these other data items influence a given coordinate only in small groups of ones or zeros, which tend to compensate for each other, i.e. white noise. The thousand copies drown out this slight noise and $\zeta$ can be successfully reconstructed.

The entire stored item is not needed to recover it. Iterated reading allows recovery when reading from a noisy version of what has been stored. Again, Kanerva offers conditions (involving how much of the stored item is available for the read operation) under which this is true, and a mathematical proof (Kanerva, 1988a). Reading with a cue that has never been written to SDM before gives, if convergent, the closest match stored in SDM to the input cue, with some sort of abstraction if close items have been written to the memory also. Since a convergent sequence of reading iterations converges very rapidly, while a divergent sequence of reading iterations bounces about seemingly at random, comparison of adjacent items in the sequence quickly tells whether or not a sequence converges.

The above discussion, based on the identity of datum and address, produced a content addressable memory with many pleasing properties. The memory is content addressable, provided we write each datum with itself as address. SDM works well for reconstructing individual memories (Hely, 1994).
3.2 SDM as a Control Structure for Autonomous Agents

3.2.1 The SDM Agent
Using SDM as a control mechanism, an autonomous agent interacts with its environment, and records its interaction. It has the potential for learning and adaptation.

Binary vectors stand for patterns of binary features. The number of features typically should be quite large.

A pattern can be used both as an address and as a datum. A sequence of patterns can be stored as a pointer chain.

Addressing the memory need not be exact. The address patterns that have been used as write addresses attract, meaning that reading within the critical distance of such an address retrieves a pattern that is closer to the written pattern than the read address is to the write address. Three to six iterations usually suffice to retrieve original patterns.

When similar patterns, e.g., an object viewed from different angles or distances, have been used as write addresses, the individual patterns written with those addresses cannot be recovered exactly. What is recovered, instead, is a statistical average or abstraction of the patterns written in that neighborhood of addresses. The object is considered to occupy a region of the pattern space with poorly defined boundaries. Thus it may stand for a concept.

The agent can rely on SDM for its internal decision on what to do next; based on the incoming percept. The agent agenda relies on the world model built in SDM (Kanerva, 1988a; 1988b).

3.2.2 Modeling the World Using SDM
Many things appear to be learned by nothing more than repeated exposure to them, i.e. learning from experience. Much of the learning process is model building. We build an internal model of the world and then operate with the model. That modeling is so basic to our nature that we are hardly aware of it.

The modeling mechanism constructs objects and individuals. A person is constantly changing and our view of him or her is different at different times, yet we perceive him or her as “that person”.

Operating with the model is like operating with a scale model. The model mimics actions and interactions of objects and individuals. The more experience we have, the more faithfully are the dynamics of the world reproduced by the model.

The model simply captures statistical regularities of the world, as mediated by the senses, and is able to reproduce them later. We can prepare ourselves for a situation by imagining ourselves in the situation.

Subjective experience produced by interaction with the outside world is of the same quality as that produced by the internal model of the world via, say, images. Our internal and external “pictures” merge without our being aware of it. We scan our surroundings for overall cues and fill in much of the detail from the internal model. However, when something unusual happens, we begin to pay attention. We are altered by the discrepancy between the external report of what is happening and the internal report of what should be happening on the basis of the past experience. Here, consciousness comes into play (Baars, 1990).

Moreover, the internal model affects our perception profoundly, without our being aware of it, i.e. prejudgments. Attention, for example, plays a great role in maneuvering our perception in terms of where to direct our perceptual apparatus, or what to pay attention to. It is immediately apparent that paying attention to something depends to a large extent on our internal model of the world and our prejudices (Foner & Maes, 1994). So, one’s internal model has a great effect on one’s ability to learn, adapt, and communicate. It is actually even one main aspect in one’s personality.

At any given moment, the individual is in some “subjective” mental state. A flow of these states describes the individual’s “subjective experience” over time. The state space for the world is immense in comparison with that for an individual’s experience.

The individual’s perceptual information at a moment is represented as a long vector of features. A sequence of such vectors represents the agent’s perception of its environment over time.

Since information created from the senses and information supplied by the memory help in producing the same subjective experience, they are both fed into some common part of the “conscious” agent’s architecture, the focus, see Figure 1.

A sequence of patterns in the focus represents the system’s “subjective experience” about the world over time. Since sequences are stored as pointer chains, the patterns of a sequence are used both as addresses and as data.

The world model is updated by writing into the memory as follows:

i. The pattern held in the focus at time \( t \) is used to address the memory, activating a set of memory locations.

ii. The response read from those locations is the memory prediction of the sensory input at time \( t+1 \).

iii. If the prediction agrees with the sensory input, there is no need to adjust the memory, and the read pattern simply becomes the contents of the focus at time \( t+1 \).

iv. If the prediction disagrees with the sensory input, a third correct pattern is computed from them, e.g. an average, and it becomes the content of the focus at time \( t+1 \). However, before it is used to address the memory at time \( t+1 \), it is written in the locations from which the faulty output was just read (the locations selected at time \( t \)).

As the correction patterns are written into memory over time, the memory builds a better model of the world, constrained only by the senses’ ability to discriminate, and the memory capacity to store information. So the agent
builds a better understanding of the world over time and is more capable of taking more rational decisions as time goes on, i.e. experience.

3.2.3 Including Action in the World Model

To act, the agent needs effectors. To learn, the agent must model the agent’s own actions. In most animals, learning to perform actions means learning to reproduce sequences of patterns that drive the muscles. Thus, the agent’s own actions can be included in the world model by storing motor sequences in memory in addition to sensory sequences, see Figure 2.

![Figure 1: Senses, Memory, and Focus in SDM. Redrawn from (Kanerva, 1988a).](image1.png)

![Figure 2: Organization of an autonomous system using SDM. Note the feedback from Motors to Sensors to allow the agent to monitor and store its own actions. Redrawn from (Kanerva, 1988a).](image2.png)
Since the way in and out of the memory is through the focus, the system effectors should be ultimately, if not directly, driven from the focus. As the system subjective experience is based on the information in the focus, deliberate action becomes part of the system’s subjective experience without the need for additional mechanisms.

Some components of the focus (70%-90%) correspond to and can be controlled by the system perceptual mechanism. Others (10%-30%) correspond to the system effectors. The focus could also have components with no immediate external significance (status and preference function). All components of the focus can be affected by the memory.

3.3 Some Psychological Aspects of SDM

3.3.1 Cued Behavior

Assume that 80% of the focus is for sensory input and 20% for effectors. Assume that the stimulus sequence <A, B, C> is to elicit the response sequence <X, Y, Z>, with A triggering action A after one time step and so on.

The pattern sequence that needs to be generated in the focus is <Aw, BX, CY, dZ>. In each pair, the first letter stands for sensory-input section and the second for effector-output section of the focus.

If <AW, BX, CY, DZ> has been previously written in memory, and A is presented to the focus through the senses, then BX is likely to be retrieved from the memory into the focus. This means that action X will be performed at the time at which B is to be observed.

Now, if the sensory report agrees with B, then BX will be used as the next memory address and CY will be retrieved, causing action Y, and so on if agreement persists.

If after BX has been read from memory, the sensory input is suppressed, the focus will be controlled entirely by the memory, and the rest of the sequence will be recalled and the actions will be completed. But, if the agent’s senses should produce the sequence <A, B, K, L> where K and L are quite different from C and D (sudden change). Then BX will retrieve CY, and action Y will be executed (inertia). Senses will report K instead of C, which was to be sensed. The next contents of the focus will be HY instead of CY (H is some combination of C and K that, in general, is quite different from C). Thus DZ will not be retrieved (failure of action Z). This failure can be explained as follows:

i. An environment monitoring system ceases to act when the proper cues are no longer present.

ii. A system that monitors its own action effects, stops acting when the effects no longer confirm the system expectations.

Since the pattern retrieved from the memory includes an expectation of the action results, the memory can be used to plan actions.

The system will initiate the “thought” in the focus and then block off the present (ignore environmental cues, and suppress action execution). The memory will then retrieve into the focus the likely consequences of the contemplated actions.

3.3.2 Learning to Act

The model’s success is judged by how well it predicts the world. When the model predicts incorrectly, it is adjusted. Action correction is much harder than perceptual correction. There is no external source feeding correct action sequences into the focus.

The action sequences have to be generated internally. They have to be evaluated as to their desirability and should be stored in memory in a way that makes desirable actions likely to be carried out and undesirable actions likely to be avoided. Learning to act means that the system stores actions in a way that increases the likelihood of finding good states and of avoiding bad ones.

4 A Cognitive Theory for Consciousness

The role of consciousness in human cognition is vital. It enables and enhances learning, allows for extra resource allocation, and deals with novel situations among other things (Baars, 1988; Edelman & Tononi, 2000). Baars’ theory of consciousness accommodates many of the features and constraints of human consciousness and cognition (Baars, 1988; 1997).

In our system, competing contexts standing for various different goals exist (Bogner et al, 2000). Players (processes represented as codelets in the system), governed by various contexts, also compete to gain access to the playing field, where they form candidate coalitions.

Only one such coalition can be in consciousness at one time. A “spotlight” shines upon the “conscious” coalition forming the “conscious experience”. So, not all the players in the playing field are in the spotlight. Only those players who are members of the “conscious” coalition are.

There is also a large audience of unconscious players waiting outside the playing field.

Once some coalition gets into “consciousness”, a broadcast of the information it carries takes place, making it accessible to everyone. This serves to recruit resources by tempting some of the audience of unconscious processes to jump into the playing field when they find something relevant in the broadcast.

Baars uses a theater metaphor to illustrate his global workspace theory. Our agent implements most, but not all of that theory. In this metaphor, conscious contents are limited to a brightly lit spot onstage, while the rest of the stage corresponds to immediate working memory, see Figure 3.

Behind the scenes are executive processes, including a director and a great variety of contextual operators, that shape conscious experience without themselves becoming conscious.
Competing for Access to Consciousness

Outer Scenes
- Seeing
- Hearing
- Feeling
- Tasting
- Smelling

Inner Scenes
- Visual Imagery
- Inner Speech
- Dreams
- Imagined Feelings

Ideas
- Imagible Ideas
- Verbalized Ideas
- Fringe Conscious Intuitions

Fringe:
Conscious Experience

Working Memory Receives Conscious Input, Controls Inner Speech, Uses Imagery for Spatial Tasks, all under Voluntary Control.

The Unconscious Audience

Memory Systems:
- Lexicon
- Semantic Networks
- Autobiographical & Declarative Memory
- Beliefs, Knowledge of the World, of oneself and others.

Interpreting Conscious Contents:
- Recognizing Objects, Faces, Speech, Events
- Syntactic Analysis
- Spatial Relationships
- Social Inferences

Motivational Systems:

Automatisms:
- Skill Memory
- Details of Language, Action Control, Reading, Thinking, and thousands more.

Figure 3: A Theater Metaphor for Conscious Experience. Redrawn from (Banks, 1997)
In the audience is a vast array of unconscious specialized processes. Some audience members are automatic routines, such as the brain mechanisms that guide eye movements, speaking, or hand movements. Others involve autobiographical memory, semantic networks representing our knowledge of the world, declarative memory for beliefs and facts, and the implicit memories that maintain attitudes, skills, and social interaction.

Elements of working memory that are on stage but not in the spotlight of consciousness are also unconscious.

Different inputs to the stage can work together to place an actor in the conscious bright spot, but once on stage, conscious information spreads, as it is widely disseminated to members of the audience. By far the most detailed functions are carried on outside of consciousness.

We need to point out here the usefulness of consciousness for humans. It is a mean for allocating resources for handling novelty and unexpected perception stimuli as well as enhancing learning. We argue that a consciousness mechanism might also be expected to be of use in a software agent.

5 CMattie, a “Conscious” Clerical Agent

CMattie is a “conscious” software agent developed to manage seminar announcements in the Mathematical Sciences Department at the University of Memphis (Franklin & Graesser, 1999). This work aims mainly to discuss the use of SDM in CMattie. For more elaborate discussions about CMattie in general, other research publications are available (Bogner et al, 2000; McCauley & Franklin, 1998; Ramamurthy et al, 1998; Zhang et al, 1998a; Zhang et al, 1998b). There was a predecessor of CMattie, namely VMattie, which lacked “consciousness” but occupied the same domain niche (Franklin, Graesser, Olde, Song, & Negatu, 1997). Both VMattie and CMattie send weekly seminar announcements via email to a mailing list that they maintain. They receive and understand email messages in natural language. This is accomplished by surface analysis. While VMattie is limited in nature in its learning capability as well as lacking some supportive modules, CMattie is augmented with a variety of cognitive structures including emotions (McCauley & Franklin, 1998), metacognition (Zhang et al, 1998a), and conceptual learning (Ramamurthy, Bogner, & Franklin, 1998).

Figure 4: CMattie Architecture (Redrawn from Bogner et al, 2000).

Almost all of the actual work of perception, action, emotions, metacognition, etc., is accomplished by codelets. These are small pieces of code, running independently, each with a relatively simple specific task to accomplish. Figure 4 contains a diagram of CMattie’s architecture.

Two main types of memory are used in CMattie, SDM as long term associative memory and a case based memory (CBM) as episodic memory, each with its own output register in the focus, see Figure 5. Both memories produce suggested actions and emotional states for CMattie. SDM takes precedence over CBM in terms of filling in default values. So, when both memories successfully read, SDM focus registers are given
precedence over the CBM registers. Each of the memories has an input and an output focus to prevent overwriting of input cues since the output from reading is often different from the input. The input focus for the two is the same. The major addition to CMattie over VMattie is a mechanism for “consciousness” in the sense of global workspace theory. This mechanism includes a spotlight controller, a coalition manager, the playing field, etc… (Bogner et al, 2000).

When an email message comes in, it gets dissected into different perception registers (PRs). Table 1 shows a list of all the PRs used in CMattie. The “NewWordOne” through “NewWordFive” fields stand for and allow learning newly defined PRs in CMattie. The niche space of the domain allow for learning new seminar concepts that need to be manipulated and stored via PRs.

Table 1: Perception Registers.

<table>
<thead>
<tr>
<th>Index</th>
<th>Perception Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SeminarName</td>
</tr>
<tr>
<td>1</td>
<td>SeminarOrganizer</td>
</tr>
<tr>
<td>2</td>
<td>SpeakerName</td>
</tr>
<tr>
<td>3</td>
<td>SpeakerAffiliation</td>
</tr>
<tr>
<td>4</td>
<td>TalkTitle</td>
</tr>
<tr>
<td>5</td>
<td>Day</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
</tr>
<tr>
<td>8</td>
<td>Place</td>
</tr>
<tr>
<td>9</td>
<td>MessageType</td>
</tr>
<tr>
<td>10</td>
<td>EmailAddress</td>
</tr>
<tr>
<td>11</td>
<td>EmergencyFlag</td>
</tr>
<tr>
<td>12</td>
<td>NewWordOne</td>
</tr>
<tr>
<td>13</td>
<td>NewWordTwo</td>
</tr>
<tr>
<td>14</td>
<td>NewWordThree</td>
</tr>
<tr>
<td>15</td>
<td>NewWordFour</td>
</tr>
<tr>
<td>16</td>
<td>NewWordFive</td>
</tr>
<tr>
<td>17</td>
<td>EmotionHappiness</td>
</tr>
<tr>
<td>18</td>
<td>EmotionSadness</td>
</tr>
<tr>
<td>19</td>
<td>EmotionAnger</td>
</tr>
<tr>
<td>20</td>
<td>EmotionFear</td>
</tr>
<tr>
<td>21</td>
<td>BehaviorName</td>
</tr>
<tr>
<td>22</td>
<td>BehaviorActivation</td>
</tr>
</tbody>
</table>

6 SDM in CMattie
6.1 Function of SDM in CMattie
SDM is used in CMattie as an associative memory performing the following functions:

i. Suggesting actions to be taken and behaviors to activate, as well as influencing the emotional status of CMattie based on the incoming percepts (values in the perception registers). Suggestion is based on previous experience, and proposes specific behaviors and emotional states in response to the observed values in the perception registers. The auto-associative SDM used here relies on the dominance, in size, of the percepts over the behaviors and emotions.

ii. Providing for defaults for missing information in perception registers, based on previous history and previous associations built over time.

iii. Removal of noise and correction of errors in percepts, based on previous history and previous associations built over time.

The incoming message is analyzed and understood and written into a group of perception registers (Zhang et al, 1998b). Those registers are then copied into the input focus for the two memories, SDM and CBM. Each memory goes into a reading cycle.

Readings may converge or diverge. Upon convergence, output from each memory is considered, with precedence given to SDM when conflict arises. The results of the reading operations are written into the output foci of SDM and CBM respectively. So we have four foci in all depicted in Figure 5.

Figure 5: Different Foci in CMattie.

The output from SDM has three distinct parts:

i. One part fills a set of registers identical to the perception registers. Defaults for missing fields are taken from here.

ii. A second contains the suggested behavior in response to the percept received.

iii. The third influences CMattie emotional status based again on the percept received.

Different modules in the architecture use various information fields, retrieved from SDM and CBM. For example, the emotional mechanism is influenced by the emotional status recommendations retrieved from SDM.
The conceptual learning module relies heavily on the output of the CBM.

6.2 Enhancements to Original SDM in CMattie

The original SDM has been enhanced in several ways to make it suitable for use in CMattie. These enhancements include:

i. Writing important or emotionally involved percepts into memory multiple times. This serves to emphasize the specific percept by giving it more repetitions internally in the memory. It also resembles a human going over emotional situations again and again in his or her mind.

ii. Using error detection and correction (Stallings 1985; Bacon and Bull 1973) to allow for original information retrieval was considered and tested. By augmenting each word with enough bits for single error detection and correction, and double error detection, we were able to remove some of the blurring of bits due to memory cluster. The problem with this approach is that special encoding for various words need to be considered to allow for enough hamming distance between various words, which makes it difficult to deal with. This feature is not included in the final version of CMattie, however, due to the added constraints imposed on word encoding.

iii. Use of hashing to encode various field values into uniform, fixed length hash keys. Then we use those hash keys for writing to and reading from SDM. For example, all various length speaker names are hashed using fixed length keys to avoid having to deal with arbitrary long names. The hash keys are actually written to and read from SDM, with the original values of the keys stored in and retrieved from a separate hash table via pre-processing and post-processing mechanisms. This approach has the advantage of uniformity but requires the extra machinery of a hash table and a hashing mechanism.

iv. Assigning weight or precedence to certain fields in the PRs, by adding extra word length for them. This serves to give more weight to the field in the semantic space, making it more capable of retrieving missing fields when used as a partial cue. The use of such weight or precedence is implementation-dependent mainly.

6.3 Tuning of SDM Parameters in CMattie

Several global parameters of SDM such as dimension, critical distance, etc., greatly affects the functionality. The capacity of SDM to store memories depends mainly on the dimension and on the number of hard locations. The convergence distance affects the degree of tolerance allowed for matches read out of the memory. Tuning these various parameters was a critical part of the development of SDM for CMattie.

The final version of SDM in CMattie uses the following parameter values:

- Address space dimension (word length): 736
- Number of hard locations: 3000
- Critical distance for divergence: 155
- Access radius for read/write: 331
- Convergence distance: 147
- Maximum number of reading iterations: 10

6.4 Run Trace

Here we include a pseudo complete trace of an email message through the CMattie system to the point at which behaviors and emotions are determined. This will provide a view of the role SDM plays in the CMattie architecture. The example email message is a request to add a name to the mailing list. Keep in mind that many of the operations described are carried out by codelets.

i. The incoming email message (sensory input) is analyzed and understood by the perception module, and moved into the appropriate perception registers (PRs). In this example, the following values might be assigned for the PRs in order:

```
```

ii. The contents of the PRs are passed to the input foci of SDM and CBM, see Figure 5.

iii. SDM and CBM go into reading cycles. Note that the behavior and emotion fields are “EMPTY” when reading.

iv. Upon convergence, the output from SDM and CBM is returned to the appropriate registers in their respective foci. Default values are thus moved into the PRs, with precedence given to SDM whenever there is a conflict with CBM. The output would typically contain appropriate suggestions for the behavior and its activation, as well as suggested values for the levels of the four basic emotions. The output in our example depends upon what has been previously written into memory. Let us assume that the SDM output behavior is “Update Seminar List Recipients” with high enough activation, e.g. 0.9, to make it execute.

v. “Consciousness” plays its role and the content of the focus, including some output from memory,
gets broadcast. This makes the information available to the behavior net as well as the emotion module.

vi. The emotion module extracts the emotion part of the broadcast (originally from the memory output).

vii. The behavior codelets extract the behavior part of the broadcast (originally from the memory output).

viii. The emotion module determines the new emotional status for the agent and puts it into the input focus overwriting the “EMPTY” placed by perception.

ix. The behavior net determines the new behavior activation for the agent and puts it into the input focus overwriting the “EMPTY” placed by perception.

x. SDM and CBM go into a write cycle to update the agent memory with the new input percept along with the corresponding emotion and behaviors.

7 Results and Statistics

7.1 Percentage of Correct Action and Emotional State
Various tests of SDM in CMattie were performed. The results shown here are the most important results obtained.

In gathering those results, tuning was accomplished involving many comparisons of various values of the implementation parameters of SDM. Those parameters include -but not limited to- word length, number of hard locations used, and critical distance.

Table 2 shows the percentage of correct behaviors and emotional states in response to a percept of sufficient length. Correctness is determined by whether the retrieved action/behavior or emotional state from SDM matches the one chosen by the behavior network and emotional mechanism respectively. We conjecture that this degree of accuracy of expectation might compare favorably with that of humans.

<table>
<thead>
<tr>
<th>Action/Behavior</th>
<th>Emotional State</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Correct</td>
<td>% Incorrect</td>
</tr>
<tr>
<td>89</td>
<td>92</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>

7.2 Percentage of Correct Default Extraction

Table 3 shows the percentage of correct defaults supplied to certain perception registers (PRs) in response to a percept of sufficient length, i.e. more than 70% of the percept is valid and noise free.

<table>
<thead>
<tr>
<th>PR</th>
<th>% Correct</th>
<th>% Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeminarOrganizer</td>
<td>82</td>
<td>18</td>
</tr>
<tr>
<td>SeminarName</td>
<td>92</td>
<td>8</td>
</tr>
<tr>
<td>Day</td>
<td>98</td>
<td>2</td>
</tr>
<tr>
<td>Time</td>
<td>91</td>
<td>9</td>
</tr>
<tr>
<td>Place</td>
<td>98</td>
<td>2</td>
</tr>
</tbody>
</table>

7.3 Effect of Memory Volume, Cue Volume, Similarity, and Noise on SDM Performance

Memory Volume is the average number of features in the memory trace. In other words, it is the average number of 1’s in an SDM word. It measures the richness of the memory trace (Loftus & Loftus, 1976). 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.

Cue Volume is the same as Memory volume but for a single input cue to SDM. 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 SDM. The more similar the words written to SDM 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. 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 (Anwar et al, 1999) a uniform distribution of the hard locations in SDM can be obtained.

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

We tested SDM performance considering the mutual effect of the four performance parameters above, using a LOW and HIGH scale, see Table 4.

The results obtained, show that the higher the volume of the memory and/or the cue length, the better the recall. However, when the memory volume is quite low, the recall gets really affected. As expected, the lower the noise, the better the recall. The more distinct the memory words are, i.e. less similarity, the better the recall, that is, the less generalization.

Note the huge difference in memory recall hit ratio in row 4, which is kind of the worst case scenario (Low Memory Volume, Low Cue Volume, High Similarity, and High Noise) versus row 13, which represents the best case scenario (High Memory Volume, High Cue Volume, Low Similarity, and Low Noise).
Table 4: Effect of Operational Parameters (Memory Volume, Cue Volume, Similarity, and Noise) on SDM Recall. Memory Volume and Retrieval Volume, H = 60%, L = 10%. Similarity, H = 70%, L = 30%. Noise, H = 30%, L = 10%. (H for High, and L for Low).

<table>
<thead>
<tr>
<th>#</th>
<th>Memory Volume</th>
<th>Retrieval Volume</th>
<th>Similarity</th>
<th>Noise</th>
<th>Hit % in Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>97</td>
</tr>
<tr>
<td>10</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>89</td>
</tr>
<tr>
<td>11</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>78</td>
</tr>
<tr>
<td>12</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>71</td>
</tr>
<tr>
<td>13</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>93</td>
</tr>
<tr>
<td>14</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>87</td>
</tr>
<tr>
<td>15</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>86</td>
</tr>
<tr>
<td>16</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>84</td>
</tr>
</tbody>
</table>

8 Conclusions

SDM has proven to be a successful tool for associative memory in “conscious” software agents. SDM is capable of recovering previously encountered percepts based on a part of that percept. It can also find defaults for empty perception registers. The evaluation of suggested actions and emotional states is satisfactory. Complete evaluation will shed more light on how well SDM is able to learn percept-action-emotion associations.

The enhancements to SDM including multiple writes of important items, use of error detection and correction, and the use of hashing to map the original information into fixed size keys were used. The use of genetic algorithms to enhance SDM (Anwar, Dasgupta, and Franklin 1999) proves quite useful for use of SDM in the CMattie domain.

The recall capability of SDM illustrated in section 7.3, shows that SDM use is sound cognitive-wise. From a computer science or AI perspective, the memory proved to be quite useful and efficient in terms of successfully retrieving the right associations for actions and emotions.

9 Future Research

One of the major difficulties encountered in using SDM as an associative memory, is its inability to recover associations based upon relatively small cues, as we humans do. For SDM to converge, a sufficiently large portion of a previously written word must be presented to the memory as an address. We are currently working on a new version of SDM that addresses this problem. If successful, this new technique should solve one of the major open problems for the use of SDM as associative memory.

Bibliography


Association for Artificial Intelligence. Stanford, CA, USA: AAAI.
Research Center, School of Computer Science, The University of Birmingham, England.

