CONCEPT LEARNING WITH NEURAL NETWORKS IN THE EVOLUTION OF A MULTI-LINGUAL UNIVERSAL KNOWLEDGE BASE

by

Eric Molas* and Winfried K. Rudloff
Governors State University

Abstract

Neural networks have the capability of learning and recalling visual and oral patterns. Specifically, artificial neural networks have been developed that seem to emulate the neural computing power of our brains. Thus, intellectual behavior such as pattern recognition, associative memory, and learning by example that is difficult to simulate with conventional means of artificial intelligence, is readily accessible to neural networks.

In our recent study on natural language processing with neural networks, we demonstrated the learning behavior of a neural network where the learning curve that evolved "live" on the computer, indicated typical learning and unlearning characteristics of the human brain (Rudloff, W. & Bengtson, M., 1990). We also proposed a binary-coded universal knowledge base that may have the potential of multilingual processing of conceptual knowledge (Rudloff, W. & Siebert, E. 1992, 1993).

This paper will discuss further developments in our neural network research. Special emphasis will be directed towards language recognition and the potential that neural networks have to process language in a conceptual framework facilitating simple translation between different natural languages. It is stipulated that understanding the behavior of artificial neural networks will eventually lead to a better assessment of how our brain processes linguistic functions and conceptualization.

Prologue: The Universal Conceptual Knowledge Base

There are many approaches to natural language processing that are rather complex, such as, Transformational Generative Grammar involving phrase structure and transformational rules, Transition Networks, Case Grammar, Semantic Networks, and Conceptual Dependency Theory. They all have in common that knowledge representation and recall are treated in a rather complicated manner. Because of such complexity, automated translation from one natural language into another has not yet reached the level of human interpreters. (Refs)

During the past years, we have thought of ways to simplify the translational process in analogy to natural neural computation of our brains. To this end, we directed our research efforts towards the development of a binary conceptual knowledge representation that may eventually be integrated into a simplified multilingual knowledge base with language-specific neural network learning and recall interfaces (Fig. 1). (Refs). For the purpose of a simplified interlingual translation, we considered the following:

1. It is estimated that there are about 3000 distinct languages on Earth (not counting dialects).

1 Excerpted, in part, from Eric Molas' Master's Thesis on “The Addition of Tag Bits in Bi-Directional Associative Memories to Achieve Multi-Dimensional Pattern Relationships”
2. Highly developed languages such as English, French or German contain approximately 200,000 words or concepts (Rudloff & Hering, 1990).

3. The evolution of natural language is intimately related to conceptual thinking. Language structures such as grammar and syntax are analytical afterthoughts that play only a role in formal linguistic context but are not necessary in understanding the content of lingual communication.

4. All human languages include primitive concepts that are universal such as, mother, father, food, water, man, woman, etc. Beyond that, global communication has brought about transference of many technological and cultural concepts between different peoples that have been integrated as loan words into their languages. These concepts are represented, through language, differently for different cultures. Obviously, a database could be created containing all of the information regarding translations from one language to a particular concept. A simple search could be done in order to look up this information. However, this approach would be extremely inefficient in that the database containing expressions for all concepts in 3000 different languages.

5. Universal concepts can be encoded in simplified binary code as is modelled in Figure 2. Here the permutation of on- and off-bits in a three-dimensional lattice can lead to the representation of large numbers of concepts. For example, an 8x8x8 binary lattice can represent $2^{24}$ or 16,777,216 distinct concepts. Figure 2 shows a smaller 4x4x4 three-dimensional matrix describing our idea of a binary conceptual knowledge base (Rudloff & Siebert, 1992).

6. Neural networks have the inherent capacity to store and recognize patterns. Specifically, they can map and recall language patterns, i.e. symbolic written as well as phonetic patterns. Figure 3 indicates how a singular concept can be mapped to expressions in diverse languages via the binary-coded universal knowledge base applying the idea of neural network-driven interfaces between the different natural languages. Language-specific coding is combined with the conceptual universal knowledge base to facilitate translation between different languages. Each language has its own specific written or phonetic symbolism that relates to patterns of alphabetic, phonetic, and hieroglyphic (pictorial) expressions of the same concept. Thus, the semantics of a concept is simply expressed by the same binary code in the universal knowledge base, while language-specificity is provided through binary-coded abstraction of letters combined to words and sentences or of pictographs.

7. Bidirectional Associative Memory (BAM) is a specific type of neural network that can be trained to recognize and recall patterns of linguistic nature (Refs). In contrast to a simple BAM, BAM systems can map to many different patterns on a one-to-one relationship. In terms of the network's energy profile, there are stable energy states (attractive basins) to which the BAM will stabilize representing some state of knowledge. Such knowledge can be recalled if patterns are properly mapped.
Bidirectional Associative Memories

Bidirectional Associative Memories (BAMs) are neural network-based constructs which attempt to achieve content addressability [Kosko 1988]. A BAM encodes information in the form of pattern pairs in a discrete additive environment. These pattern pairs can be realized in any real-valued matrix. A BAM is bidirectionally stable, as was shown by Amari [1977] and Hopfield [1982]. Thus, there exist stable energy minima called attractor basins, to which the BAM will equilibrate. Figure 4 shows such minima. Retrieval will always occur when presented an input stimulus. However, this does not indicate which item in memory the pattern recalls.

In a BAM system in which there is a complete one-to-one relationship between all pattern pairs presented (disregarding ambiguous information), the BAM system will recall the trained output pattern in constant time. On the other hand, in a traditional BAM, if there exists a many-to-one relationship between any sets of pattern pairs, a problem occurs. If the pattern of lesser dimension (i.e. the pattern that many patterns are mapped to) is presented for recall, then only the nearest attractor basin is chosen as the analogous output. That is, you cannot recall all the patterns to which the concept is associated. A BAM has the property that information may be recalled in constant time once it has been encoded. Traditional search algorithms could not match this performance given the huge database of information as described above, ruling out such possibilities as a hash table.

This paper describes an investigation into a method called TAG BIT ADDITION in which the problem of proper mapping can be dealt with, in order to simulate this multi-dimensional relationship in a BAM. An alternative algorithm for training and recall of BAMs will be proposed. This algorithm centers around the addition of extra tag bits to the pattern of lesser dimension which allows for pseudo-parallel association.

Representing concepts as binary encoded patterns lends itself well to a neural-network based approach, because of its natural binary vectorized structure. If we wished to represent 200,000 different concepts, then our output layer would need to be at least \( \log_2(200,000) = 17.6 < 18 \) bits (or neurons). However, this number is insufficient in many-to-one mapping situations. This deficiency was first noticed when we investigated a BAM system, a spell checker (Blum, 1992). In this application, an input pattern, is a spelling of some word, and an output pattern of a phonetic representation of the word are presented to the BAM system. After all the input-output pairs have been encoded, the user would supply a word as input. The spell checker would then return a phonetic representation of the word, and a possibly corrected spelling. However, when the knowledge base was modified so that two words had the same phonetic representation, such as "there" and "their," then a flaw in the BAM immediately show up. When the BAM was confronted with the phonetic representation of "there" and "their" for recall, only one spelling was returned! The BAM did not have the ability to recall both words which were mapped to the same phonetic representation. This behavior is due to the fact that the BAM resonates to the first stable basin that it finds. Thus, since the two words "their" and "there" are slightly different, only the word which together with the binary representation of the phoneme have lowest energy, will be chosen. Yet it is central to our Universal Language Translator that it be able to handle this many-to-one relationship because several words from different languages would all point to the same binary concept pattern. Without the modification, the BAM would not have the ability to recall all of the correct translations that a user might request.

In order for our BAM system to understand textual input, we must transform it into a form which can be understood by the BAM, i.e. binary code. Since there are 26 letters of the alphabet, and \( 2^5 = 32 > 26 \), five bits (neurons) are necessary for each character of input. In addition, if we allow a maximum string length of 20 characters per concept, our input layer will be of length \( 20 \times 5 = 100 \) neurons.
Example Mappings

In the situation represented by Figure 5, a simple one-to-one relationship exists between the input and output layers. In contrast, the situation represented in Figure 6 has several words in different languages mapped to the same binary pattern. If any of the inputs is presented, then the correct binary encoding is recalled. However, in order for the Language Translator to function properly, it must be able to recall all input patterns when the binary encoded output seeks the input(s) which map to it. In the traditional BAM, only one input would be recalled (the first attractor basin which the recall algorithm found in terms of the lowest orthogonal matrix energy). Therefore, our Language Translator would not work correctly. For example, if one submits "mother" to our Universal Language Translator, then all of the analogous words meaning "mother" in other languages must be recalled. Unfortunately, in our primitive system, upon the input of "mother", only one equivalent translation will be chosen (and it might even be "mother" again!)

Traditional Encode and Recall

Encode

In order to encode a pattern pair into a BAM, one must add the matrix which is formed by the multiplication of the two vectors to be encoded to the current existing matrix. If the pattern pair to be encoded is the first pattern pair, then the starting matrix is simply a zero-filled matrix of size NxP, where N and P are the lengths of the two vectors of the pattern pair to be encoded. Figure 7 represents the operation and resultant matrix where the pattern pair of the word "mother" and its corresponding universal concept has been encoded. Note that all zeros are converted to -1’s so that there exists no bias favored toward positive 1. If the zeros were used, then each binary (0+1) operation would result in 1. This is obviously biased! This preference of bipolar over binary vector representation can be seen in the additive matrix operations that we will be continuously performing throughout the training of a pattern pair. In such a (-1+1) operation, the result is zero, in which further encoded information or a threshold value can initiate the decision to "fire or not."

The following encoding of "mother", as a pertinent example, could be converted to be represented by the matrix in Figure 7:

\[
\begin{bmatrix}
1 \\
-1 \\
1 \\
1
\end{bmatrix}
\begin{bmatrix}
-1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
-1 & 1 & 1 & 1 \\
-1 & 1 & 1 & 1
\end{bmatrix}
= 
\begin{bmatrix}
-1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
-1 & 1 & 1 & 1 \\
-1 & 1 & 1 & 1
\end{bmatrix}
\]

Figure 7: Matrix Operation of Converted (Non-Biased) Matrix

<table>
<thead>
<tr>
<th>Mother</th>
<th>01101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food</td>
<td>01110</td>
</tr>
<tr>
<td>Sleep</td>
<td>11100</td>
</tr>
</tbody>
</table>

Figure 5: Mapping of Simple Concepts

<table>
<thead>
<tr>
<th>Mother</th>
<th>01101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madre</td>
<td>01101</td>
</tr>
<tr>
<td>Mater</td>
<td>01101</td>
</tr>
</tbody>
</table>

Figure 6: Multi-Lingual Mappings to the Same Concept

\[
A_1 = 1011 \quad (Mother) \quad B_1 = 01101 \\
A_2 = 0010 \quad (Food) \quad B_2 = 01110 \\
A_3 = 1001 \quad (Madre) \quad B_1 = 01101
\]

with the sample mappings:

\[
A_1 \leftrightarrow B_1 \quad A_2 \leftrightarrow B_2 \quad A_3 \leftrightarrow B_1
\]
BAMs have an inherent information storage capacity based on the matrix dimensions (n,p). Specifically, a maximum of m pattern pairs may be encoded in an nXp-sized matrix as long as 

\[ m < \min(n,p) \]

holds true.

That is, as we are encoding our pattern pairs, we must test each pair to see if it can be properly recalled. If not, then we must erase that association from the resultant matrix by means of subtracting the currently tested pattern pair. Then another matrix is tested until one is found in which the pattern can be correctly encoded and recalled. If such a matrix is not found then we must create another information matrix. This method for creating multiple BAM matrices to overcome the inherent capacity limitation of simple BAMs is known as the BAM SYSTEM [Simpson 1990]. For the purpose of this paper, the BAM SYSTEM is adopted as the basis for our modified encode and recall routines.

Recall

After all pattern pairs have been properly encoded into the BAM system, we wish to recall any pattern \( B_i \) each time \( A_i \) is presented, for the trained pattern pair \( (A_i, B_i) \). This result vector is then passed through some sort of nonlinear threshold function chosen by the network designer. A typical threshold function might be

\[
f(x, y) = \begin{cases} 
1 & \text{if } x > 0 \\
y & \text{if } x = 0 \\
0 & \text{if } x < 0.
\end{cases}
\]

Then, the pattern which is output by the threshold function performed on each neuron is fed back in a reverse fashion through the transpose of the encode matrix to obtain another input pattern \( A_j \). As this two-step process is repeated a number of times, the input and output patterns \( A_i \) and \( B_i \) will resonate to a steady pattern. BAMs have the property of global stability, which means that this feedforward/feedback process won't go on forever. The Lyapunov Energy function is characteristic for a BAM:

\[ E(A, B) = -AWB^T. \]

where A and B are the input-output pattern pairs and W is the current BAM matrix [Blum 1992]. Thus, given that two language-specific input patterns \( A_i \) and \( A_j \) are unique in that their ASCII representations are unique, then one of them will have a lower energy when presented to the Lyapunov energy function. Lyapunov showed that any function which is zero at the origin and has non-increasing changes is globally stable. It can be shown that the Energy function for a BAM does indeed have these properties (Kosko 1988).

Building upon basic BAM recall, a multi-matrix BAM System recall is somewhat more complicated. The BAM system extends the BAM by allowing for additional matrices once saturation has occurred within one. One should remember that there is an inherent limit to the number of pattern pair associations that can be stored in a given matrix. This limit is \( \min(n,p) \), where n and p are the input/output pattern sizes. Because the encoded pattern pair for a given input vector can be located within any existing matrix, they must all be checked. Therefore, a pair \( (X_i, Y_i) \) is chosen from each matrix by presenting pattern A. The returned pattern pair which has the energy closest to the matrix's orthogonal energy, -np, is chosen. The energy of a pattern pair is as mentioned above, -AWB^T. This means that some pattern pair will be chosen from each matrix, due to the property of BAM stability. However, the correct one will be chosen by its proximity to the matrix's orthogonal energy, -np. This is ultimately how a specific pattern pair is chosen in BAM recall.

Even though the BAM system allows us to exceed the storage capacity \( \min(n,p) \), the encode and recall are simply extensions of the simple BAM equivalents. The inability to recall all output patterns in a many-to-one relationship is evident. The pattern pair with the energy closest to the matrix energy, -np, is chosen. For unique pattern pairs \( A_i . A_j \rightarrow B_k \), only one A will be chosen when \( B_k \) is presented for recall. In the following, the modified Encode and Recall which can handle this problem will be discussed.
Tag Bit Addition

The basic idea behind my modified BAM is to extend the dimensionality of the lesser-dimensional pattern by two separate groups of tag bits, each of which has a different purpose. The first group is of size 

\[ \text{ceiling} \left( \lg(N) \right) \text{ bits}, \]

where \( N \) is the maximum number of patterns that can be mapped to one particular pattern. For example, if we were dealing with 5 different languages such that:

\[ M_1 \ldots M_5 = \text{(mother, mama, madre, mutter and mater)} \]

where the "mother" concept has the binary representation \( 01101 \), then \( N \) would be five and our increase of tag bits would be:

\[ \text{ceiling} \left( \lg(5) \right) = 3. \]

Consequently, the binary representation of the concept "mother" would be increased in length by 3 bits. The inherent problem that was discussed earlier is that when \( M_1 \) through \( M_5 \) are mapped to the same concept, we can only recall one pattern \( M_i \) when presenting \( 01101 \). This means that, when a user asks in English for all other translations of the English word "mother", only one translation is given (which could be the English "mother" again!) In a Simple BAM this is demonstrated as follows:

1) English "mother" is presented as 1/2 of pattern pair for recall.
2) The mother concept \( 01101 \) is returned.
3) Mother concept \( 01101 \) is presented to BAM system for recall.
4) A pattern pair whose energy is nearest to \(-np\) is chosen. There will only be one pattern pair chosen due to the uniqueness of the pattern pair energy \( E(A,B) = -AWB^T \).

In order to overcome this problem, we propose that the binary encoding of the concept pattern be assigned uniqueness bits which give it uniqueness within its class (Figure 9). For our example above, the mother concept \( 01101 \) would be granted extra bits as follows for the five unique mappings to it (Figure 10). Each new encoding is representing the uniqueness of the mapping from a particular input. However it is easy to recall all output for a given pattern class input (i.e. the class \( 01101 \)). Simply Recall all existing mappings \( 01101xxx \). These bits would be assigned a value during encoding which is a running total of all patterns that matched to a particular pattern.

At this point, we recall that the binary encoding of the language-specific half of the input pattern is simply an ASCII representation of the character string. For the specifications of the Universal Language Translator, this layer had a neuron (bit) size of 100 (max), allowing for 20 characters, each character's ASCII code represented by 5 bits. Allowing for 200,000 concepts as described earlier would increase the concept pattern size from 18 bits to 21 bits, assuming only five valid mappings (corresponding to five different languages) were allowed to each concept. However, we would be increasing this pattern size only logarithmically, not changing the general matrix size greatly even if many more mappings were allowed to a single concept pattern.

The second group of tag bits will be used in keeping track of how many total mappings have been made to a particular concept. Another ceiling \( (\lg(N)) \) bits would have to be added at the tail end of the concept pattern. The information contained within these bits would be the total number of recorded
mappings to that concept. For example, given our previous example of Figures 9 and 10, we have five language specific patterns mapping the concept pattern "mother" 01101 (See Figure 11).

The value of five in the tail end of the encoding will be used during recall in order to be able to retrieve only valid mappings to a concept, and no more. How this number is found and encoded will be discussed in detail in the description of the Modified Encode.

It might seem from this example that the number of tag bits added is dynamical. However, in order for the BAM system to function properly, the input and output layers must each have their own constant size. Therefore, the BAM system designer must have an idea as to how many different input (language-specific) patterns may map to a single concept pattern before any training is done. This maximum number allowed for any pattern pair, N, determines the amount of additional Uniqueness bits and STOP tag bits, even before we start presenting training pairs!

In our example, our five phrases for the concept "mother" would have the tag bits 001, 010, 011, 100, and 101 added on to their original encodings. Thus, each has a unique encoding, while still maintaining class equivalence. In the following, we will discuss the second set of tag bits, the STOP bits, and their function.

### The Difficulty in Keeping Running Totals

The goal of the BAM system is to achieve content addressability. Information can and should be recalled in constant time given it has been properly encoded. The problem occurs when trying to assign the running total value for the tag bits of a particular concept. How is the BAM system to know how many associations have already been made to a particular concept? The information contained in a BAM system is the matrices resulting from the sums of the individually multiplied input/output pattern pair vectors, as described in Figures 7 and 8. Thus, when a particular pattern pair is presented for encoding, how is it to know that it is, say, the third pattern pair which maps the same concept location? Unfortunately, the solution is not computationally easy.

Assume an input/output pattern pair (A,B), which maps a language specific pattern A to a concept B. The binary encoded representation of B does not yet contain any tag bits. The BAM system designer has already decided on and placed a maximum limit on the maximum number of mappings allowed to a concept, N. In order to determine how many patterns have mapped to a particular concept, all of the N permutations of the concept class whose base has binary encoding B must be checked for existence.

For example, if B were the concept "mother", which had an encoding of 01101, and N was decided to be 8, then the first set of tag bits, the uniqueness bits, would be added on. For example, 01101000, 01101001, ..., 01101111 would be the intermediate patterns resulting from this first step. Secondly, the stop bits would be added on after this. The stop bits can be predetermined before any training occurs by simply running a scan on the fact file and by incrementing a running total for each mapping found to a particular concept pattern. Later on, when the number of mappings to each concept pattern is tallied and stored, these stop bits will be filled with the stop value. This stop value for our example would be five. I will show its use now in a complete example.

Say that the maximum number of mappings from a language-specific pattern to a universal concept pattern were decided to be N = 8. We seek to teach the modified BAM the five facts M₁..M₅ all mapped to 01101, the "mother" concept B, as shown in Figure 7.

\[ M₁..M₅ = \{\text{mother, madre, mutter, mater, mama}\} \]

The 5 pattern pairs (facts) are trained in the order:

\[ M₁ \rightarrow B, M₂ \rightarrow B, ..., M₅ \rightarrow B. \]

It was also determined, by scanning the fact file, that there were a total of 5 mappings to the mother concept B. Therefore the stop bits for these five mappings are 101.

The concept mapping encoding will be (for these five facts):

\[ 01101 + xxx + 101 \]

with xxx being the uniqueness bits, ranging from 000 to 101.
When the first fact is encoded, since no other mappings have been made to 01101, B is given a uniqueness tag of 000. The total number of mappings to 01101 (currently 1) is incremented and stored in some table location for 01101. So the concept pattern mapped to it is 01101000101. When the second fact is trained, all possible N unique permutations of 01101 are checked for recall. Let's call the N language patterns recalled A1..AN. A1..AN are then checked on the already trained fact file for existence. In our case, the only one found would be the first pattern "mother". The uniqueness value is the highest uniqueness value so far, plus one. Therefore 01101001101 is the concept pattern for this mapping. This process is repeated for all facts.

**Computational Costs**

Obviously this encoding process is computationally intensive. Recall would have to be performed N times, the maximum amount of mappings to a concept, for the training of a single fact. Also, some sort of search would have to be performed on the fact file for tallying the mappings to each concept. However, these initial costs are included only in the training phase of the BAM when information is being entered. The constant-time recall for the user is still preserved. In the following, he algorithm for the Modified Encode is finally presented.

**Modified Encode Algorithm**

Given:
- N is the maximum allowed mappings to a particular concept from different language specific patterns.
- T is the table which contains the count of mappings to each concept pattern contained in the fact file.
- Uniqueness Tag bits e1..elg(N).
- Stop Tag bits f1..flg(N).
- Language-specific patterns A are of size n in \{0,1\}.
- Concept patterns B are of size p in \{0,1\}.
- Input/Output pattern pair mappings A \rightarrow B (facts) stored in some fact file F.
- A set of BAM matrices M1..M_X of size (n,p+\log2(N)) in which to store our pattern pair information.

The resulting algorithm would be as follows:

1. initialize-matrix(M)
2. tally(F,T)
3. for (each fact A \rightarrow B of F) do
   4. MAX ← -1
   5. f1..flg(N) ← get_tally(T,B1..Bp)
   6. for (j = 0 to N-1) do
      7. e1..elg(N) ← j
      8. Aj ← recall(B1..Bp,e1..elg(N)f1..flg(N))
      9. if (Aj exists in already trained portion of F) then if |e1..elg(N)| > MAX
         10. MAX ← |e1..elg(N)|
      11. if (MAX+1) > N then "error"
      12. else e1..elg(N) ← MAX+1
      13. traditional-BAM-encode(A,B1..Bp,e1..elg(N)f1..flg(N))

Line 1 simply initializes the matrix M(n,p+\log2(N)) to zero. Line 2 runs the tally procedure which sums all the the mappings to each unique concept pattern in F and records this in a table T. Line 3 loops through each pattern pair of the fact file. MAX of line 4 holds the number of associations already mapped to the concept pattern B. It is initialized in line 4 to -1. The stop bits are assigned their proper value in line 5. Line 6 loops N times, once for each possible mapping to the current concept pattern B. Line 7
assigns the tag bits a value. The value will range from 1 to N-1. Line 8 returns a language-specific pattern \( A_j \) for each of the N permutations of B. In line 9, the fact file F is checked to see if a particular pattern \( A_j \) exists. This is a validity check, because some pattern \( A_j \) will always be recalled by the call to recall in line 8. This is a property of a BAM in that some (possibly incorrect) association can always be recalled. Line 9 assures that \( A_j \) was a previously trained pattern. If the validity check passes in line 9, then we possibly increase \( MAX \) to the current uniqueness value. Eventually \( MAX \) will hold the highest absolute value of all of the valid uniqueness bits checked (lines 10, 11). Line 12 is an error trap to catch any possibilities for exceeding the maximum allowable amount of mappings to a concept pattern. Otherwise, we assign the next available uniqueness tag bits to perform the standard BAM System encode in lines 13 and 14.

**Modified Recall**

My goal has been to be able to retrieve all translations for a given input language-specific pattern. Many different languages share common concepts. Thus was born the need to properly recall all language phrases when a given concept is presented for recall.

Given: N is the maximum allowed mappings to a particular concept from different language specific patterns.

T is the table which contains the count of mappings to each concept pattern contained in the fact file.

Uniqueness Tag bits \( e_1..e_{\lg(N)} \) for each concept pattern B.

Stop Tag bits \( f_1..f_{\lg(N)} \) for each concept pattern B.

Language specific patterns A are of size n in \{0,1\}.

Concept patterns B are of size p in \{0,1\}.

Input/Output pattern pair mappings \( A \rightarrow B \) (facts) stored in some fact file F.

A set of BAM matrices \( M_1..M_X \) of size \((n,p+\log_2(N))\) in which to store our pattern pair information.

The Modified-Recall returns a set of language-specific patterns \( A_1..A_{STOP} \), where STOP is the magnitude of a particular concept pattern's Stop Tag bits \( f_1..f_{\lg(N)} \). Following is the algorithm of the Modified Recall:

1. \( B \leftarrow \text{traditional-recall}(A) \)
2. \( f_1..f_{\lg(N)} \leftarrow \text{get-STOP-bits}(B) \)
3. \( \text{STOP} \leftarrow \text{get_magnitude_of}(f_1..f_{\lg(N)}) \)
4. \( \text{for } (j = 0 \text{ to STOP-1}) \text{ do} \)
5. \( e_1..e_{\lg(N)} \leftarrow \text{convert-to-bit-string}(j) \)
6. \( A_j \leftarrow \text{traditional-recall}(B_1..B_p^e_1..e_{\lg(N)}f_1..f_{\lg(N)}) \)

Line 1 retrieves the concept pattern for a given language-specific pattern. Line 2 truncates the STOP-bits from B and stores them in the bit array \( f_1..f_{\lg(N)} \). In line 3, the integer variable STOP is given the value of the magnitude of the stop bits. This will allow us to only retrieve the patterns which have been encoded (there will be the number STOP valid patterns out of a total of N possibilities, where N is the maximum amount of mappings to a concept allowed). Lines 4, 5, and 6 loop STOP times, placing the uniqueness bits in each recall according to what the current loop iteration is.

All of the translations of the input pattern A are stored in the array of patterns \( A_1..A_{STOP} \). Some further work could be done in order to encode a "language identifier" into the BAM system. This could allow for the user to mask out a certain set of languages that he/she was seeking translations for.

**Conclusion**

Through the addition of two sets of tag bits, *uniqueness bits* and *stop bits*, we have enabled a Bidirectional Associative Memory System to recall complicated relationships between input and output pattern pairs. In terms of the Universal Language Translator, it would be impossible to recall all of the many different language patterns which all mean the same thing, and thus map to the same concept pattern. However, the lists of pattern pairs to be trained on the BAM system must be scanned a-priori in order to
find out how many mappings there are to each unique concept. This information is used and part of the encoding of the concept pattern as the STOP bits. There could be N possible mappings to a particular concept, but the STOP bits contain the exact amount to a concept. The uniqueness bits are necessary because the BAM system demands unique relationships between any input/output pattern pair in order to be recalled correctly. However, every set of mappings to a particular concept still has the same base class for identification.

The running time of encode is large in that recall must be performed N times for each pattern pair to be encoded in order to determine the next available uniqueness bits to assign to a concept for that particular association. However, once this information is determined and the pattern pairs are trained, then recall still runs in constant time, which is the visible running time of a BAM system when it is in use. The space necessary for our modified BAM is increased from traditional versions. The size of the BAM matrices are extended from (n,p) to (n,p+2log₂(N)). The number N is determined by how many different mappings will be allowed to a particular concept. For the Universal Language Translator, this depends on how many languages will be included, and how many phrases from each language mean the same thing. The increase in size of the binary matrices is, however, only logarithmic.

Further research can be done in several areas concerning this project. Firstly, other possible kinds of tag bits can be considered for addition. For example, it would be possible to add a language-specific set of tag bits so that the user could choose which languages should be returned. Thus, when the concept pattern is found from the presenting an input pattern, a recall could be performed with the input pattern being the concept pattern with the language bits changed to the user requested value. Also, the project could be tied in with a voice recognition system which could translate voice input to text, and have that text used as input to the Universal Language Translator.

REFERENCES


Rudloff, W. K and E. Siebert, "Integration of Neural Networks with a Universal Knowledge Base: An Exercise in Research with Undergraduate and Graduate Students", paper presented at the 4th International Symposium on Systems Research, Informatics, and Cybernetics, August 1993, Baden-Baden, Germany.


