Learning Recursive Distributed Representations for Holistic Computation

Lonnie Chrisman

July 1991
CMU-CS-91-154

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15217

Abstract

A number of connectionist models capable of representing data with compositional structure have recently appeared. These new models suggest the intriguing possibility of performing holistic structure-sensitive computations with distributed representations. Two possible forms of holistic inference, *transformational inference* and *confluent inference*, are identified and compared. Transformational inference was successfully demonstrated in [Chalmers, 1990]; however, since the pure transformational approach does not consider the eventual inference tasks during the process of learning its representations, there is a drawback that the holistic transformation corresponding to a given inference task could become arbitrarily complex, and thus very difficult to learn. Confluent inference addresses this drawback by achieving a tight coupling between the distributed representations of a problem and the solution for the given inference task while the net is still learning its representations. A dual-ported RAAM architecture based on Pollack's Recursive Auto-Associative Memory is implemented and demonstrated in the domain of Natural Language translation.

This research was sponsored by NASA under contract number NAGW-1175. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the U.S. government.

Keywords: Artificial intelligence, learning, connectionism and neural nets, structure-sensitive holistic computation, confluent inference, natural language translation.

Learning Recursive Distributed Representations for Holistic Computation

Lonnie Chrisman

1 Introduction

It is generally agreed upon that many cognitive tasks require the use of data containing combinatorial constituent structure. Classical examples of such structure include graphs, trees, and lists. The inadequacies of most connectionist models of not being able to represent or make use of such structure has limited the application of these models to higher level cognitive tasks and has been a source for attacks on the connectionist enterprise [Fodor and Pylyshyn, 1988]. However, recently several connectionist models with the capability for representing structured data have been introduced ([Pollack, 1990], [Elman, 1990b], [Smolensky, 1990], [Hinton, 1990], [St. John and McClelland, 1990], [Miikkulainen and Dyer, 1989], [Lee *et al.*, 1990]). These usually map syntactic compositional structure into distributed representations by using various composing and decomposing functional operations.

The emergence of these new distributed representations for structured data creates the possibility for a new and intriguing mode of computation called *holistic inference*. This form of inference occurs in a *gestalt* fashion by deriving a solution directly from the representation of structured data without decomposing, locating, or accessing its constituent elements. The most interesting case revolves around distributed representations that [Van Gelder, 1990] characterizes as possessing *functional compositionality* without *concatenative compositionality* — ie. representations where the elements or relationships between elements are not easily ascertained from the surface structure. Such holistic inference is only feasible as a result of *micro-structure* that emerges in the representation. This micro-structure may provide a means for the extremely efficient computation of certain classes of inference such as what [Hinton, 1990] refers to as *intuitive inference*.

It is far from obvious that structure-sensitive holistic inference could even be possible. If a distributed representation is viewed as a complicated encryption of the original data, then there is no reason to believe that such a representation might directly reflect any pertinent content. On the other hand, the touted abilities of neural nets to capture relevant regularities in data may instead be directly reflected in the resulting micro-structure of the representations, thus opening the doors to a whole new realm of inference, the essence of which would be radically different from most people's conceptions of how computation must be performed.

Chalmer's fascinating experiment [Chalmers, 1990] gives the first positive indication that such structure-sensitive holistic inference is, in fact, possible. Distributed representations were derived for a corpus of sentences using Pollack's Recursive Auto-Associative Memory (RAAM) architecture. A simple *transformation network* was successfully trained to perform sentence passifization (ie. converting a sentence such as "John loves Michael" from the active into the passive, "Michael is loved by John") by mapping directly from distributed representation to distributed representation. On a corpus of 75 active-passive training pairs and a different set of 75 active-passive testing pairs, the transformation network achieved an amazing 100% accuracy, thus giving an existence proof of structure-sensitive holistic computation.

In the general case, the use of pure transformational inference, as employed by Chalmers, is subject to significant transformational complexity which can easily hinder the success of holistic inference. The drawback arises because the (distributed) representations are learned independently of the inference tasks for which they will be used. As the representations for the input and output of a given inference task become increasingly disparate and nonsystematic, the complexity of the transformation necessary to encode the inference grows significantly. The resulting complexity can critically impede the effectiveness of the holistic transformation.

This paper introduces a method, termed *confluent inference*, for addressing the difficulty of transformational complexity. The basic idea is to account for the inference task(s) while learning representations by encouraging the confluence of the representations for the problem and answer of the inference task. While confluent inference should properly be viewed as part of the representation learning process, *in the extreme* it is a novel form of structure-sensitive holistic inference that can accomplish the entire inference task by itself. As a representation forming mechanism, confluent inference can act synergistically with transformational inference. To explore the abilities of the pure confluent approach, a dual-ported RAAM architecture was devised and implemented and applied to a small English $\leftrightarrow$ Spanish translation domain.

The paper begins by reviewing the basic RAAM architecture [Pollack, 1990]. The two types of holistic computation, transformational and confluent, are then presented in detail and compared. Next, the dual-ported RAAM and the associated training technique are developed, and experimental results from applying the architecture to a natural language translation task are given. The help define the scope of applicability, some possible variations on inference tasks are considered, and a discussion of the conditions that allow confluent inference to be effective follows. Finally, methods are presented for synergistically combining confluent and transformational inference.

2 RAAM

Pollack's Recursive Auto-Associate Memory (RAAM) architecture allows variable-sized structured data to be representing using a fixed-sized network [Pollack, 1990]. The basic RAAM can encode arbitrary tree structures of variable depth as long as the valence (branching factor) is bounded. In theory, there is no hard limit placed upon the maximum depth of any branch, nor is there any specific upper bound on the number of distinct trees that can be stored. Of course in practice, the maximal depth and number of trees that can be stored and retrieved accurately depends upon the network's capacity, as determined by its size. When representing an arbitrary tree, the basic RAAM requires that the number of units used to represent a terminal element be equal to the number of hidden units used to represent a complete data structure. In the special case of a list, this restriction can be lifted and the resulting configuration is called a *Sequential RAAM*. For simplicity, the description here will be limited to the Sequential RAAM, but all methods discussed in this paper generalize straightforwardly to the basic RAAM.

The encoding and decoding of a list can be accomplished using the recursive configuration shown in Figure 1. After training is complete, a list can be encoded by placing a local representation for the first list element on the left L units of the input, and an *empty* or `nil` vector on the right K units. This produces a distributed representation for a list of one element on the K hidden units. The activations of the K hidden units are then copied to the rightmost K units of the input and the second element is placed on the left L units, producing a representation on the K hidden units for the two element list. The process is continued until the entire list is encoded.

A list can be decoded by placing its distributed representation directly on the hidden units. The leftmost L units of the output return the last element of the list, while the rightmost K units return the representation for the remainder of the list. The decoding process can be repeated until the end of the list is detected, for example, by detecting a non-terminal element in the leftmost L units of the output[1].

The network is trained to *auto-associate* the desired inputs by using the back-propagation procedure. A list element, plus the encoding of the proceeding portion of the list, are placed on the input units and the network is trained to reproduce this same pattern on the output units. In the process, the network is forced to develop a compressed representation on the hidden units. The hidden activations are extracted, used to encode longer lists, and back-propagation is repeatedly applied until the end of the list is reached. As the network learns, the the hidden unit encoding changes, and a form of *moving target* learning emerges.

After this process is carried to completion, we are left with both an *encoding* process and a *decoding* process since the output layer of the net can be used to decode a list as described earlier.

3 Holistic Computation

The RAAM architecture described in the previous section can be viewed simply as a distributed memory for storing compositional data structures. Viewed in this way, computation proceeds by locating, extracting, and combining constituent elements of the encoded structures in a manner not much different from traditional symbolic processing. Nonetheless, various emergent properties of distributed representations [Hinton *et al.*, 1986] and of the RAAM architecture (such as high encoding efficiencies, fault tolerance, or the tendency to make mistakes gracefully) may make this view of structure-storing connectionist models interesting in their own right. The best example of such use is BoltzCONS [Touretzky, 1990].

But beyond being just a structure storage device, the RAAM's representations suggest

[1] A simple threshold τ can be set to detect non-terminal elements. For example, if any output unit o_i has an activation level $o_i > \tau$ or $o_i < 1 - \tau$, the token is considered to be a non-terminal. Usually τ is set to 0.2.

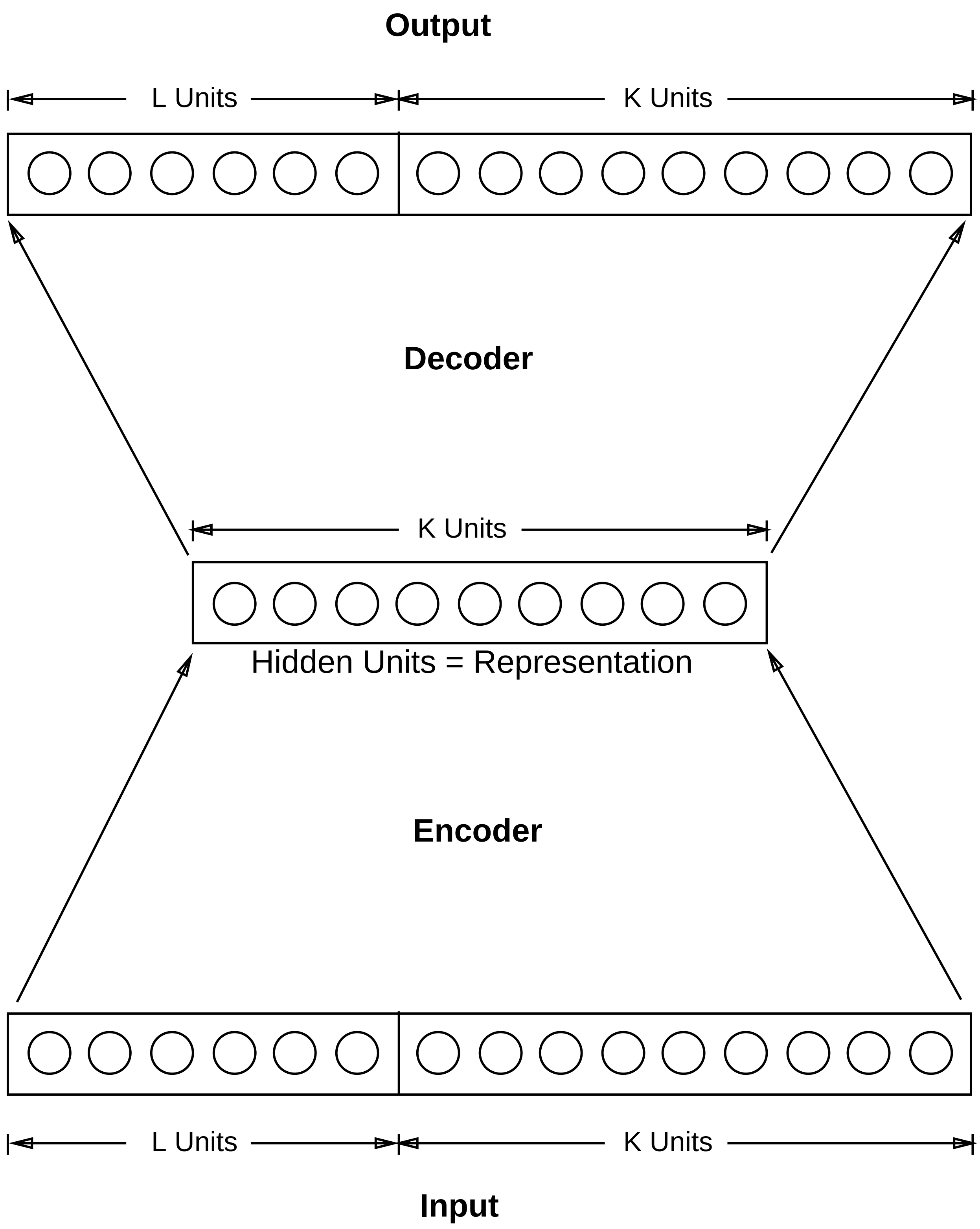

Figure 1: The Sequential RAAM Structure.

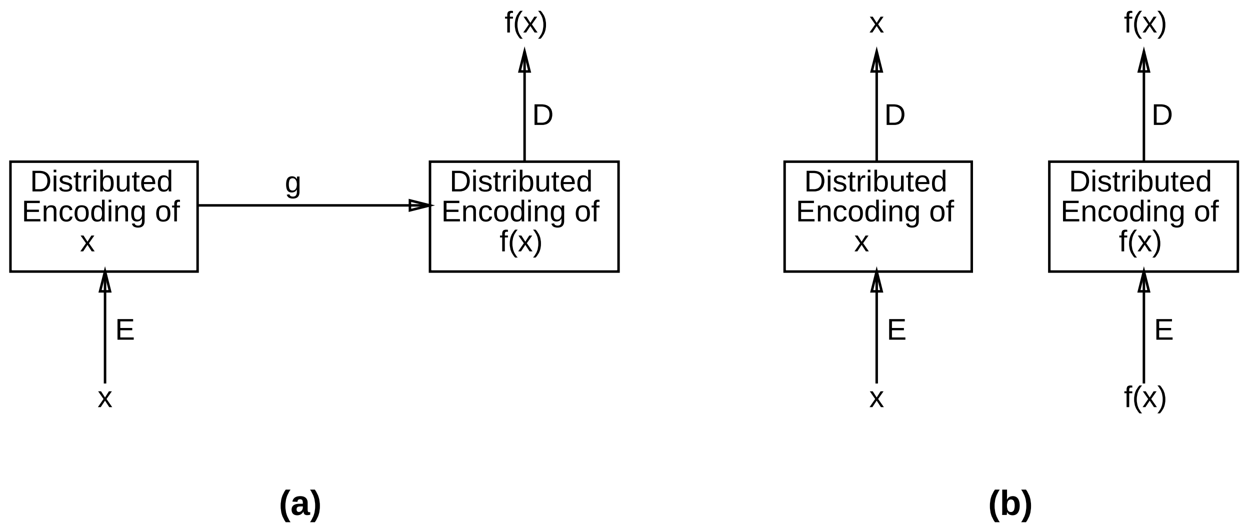

Figure 2: (a) The transformational holistic computation of $f(x)$ by $g(\cdot)$. (b) Usually the auto-association of x and $f(x)$ is used to learn representations before learning the transformation $g(\cdot)$.

that much more may be possible. In order to compress arbitrary data structures down to a fixed-width hidden layer, the RAAM must make use of regularities, and these may become reflected directly in the distributed representations. These representations encode all the information from the original data structure as well as an additional statistically-based *microsemantics* between constituent elements and relations within the data. Thus, highly efficient computation that takes advantage of this otherwise unavailable microsemantics might use the distributed representations directly. In the terminology of [Van Gelder, 1990], the distributed representations learned by a RAAM possess functional compositionality without concatenative compositionality. Thus, the individual elements of a data structure, and the relationships between these elements, are not usually directly reflected in the resulting representation. Inferences that use only the surface micro-structure of a representation without accessing its compositional structure is said to perform *holistic inference.*

3.1 Transformational Inference

Let x represent a given item of structured data. The encoding process of a RAAM maps $x \in X$ to a distributed representation denoted by $E(x) \in R$. Similarly, the decoding process maps a representation $r \in R$ to a data structure denoted by $D(r) \in Y$. When $D(E(x)) = x$, we say the network is capable of auto-associating (ie. representing) x. Recall that the computation of $E(\cdot)$ or $D(\cdot)$ by a RAAM requires multiple encoding or decoding steps.

We can view a given inference task as computing a function $f : X \rightarrow Y$, where elements of X and Y are structured data. For example, X may be the set of all English language sentences, Y the set of all Spanish sentences, and $f(\cdot)$ is the translation function that converts any sentence from English to Spanish.

When $\langle E(\cdot), D(\cdot)\rangle$ exhibits *functional compositionality* without *concatenative composi-*

tionality, and when $D(g(E(x))) = f(x)$, then $g(\cdot)$ performs a *transformational holistic computation* of $f(\cdot)$. The process of computing $f(x)$ in this fashion is called *transformational holistic inference* and is diagramed in Figure 2(a). This characterizes the techniques employed in [Chalmers, 1990] and the syntactic transformation experiments of [Blank *et al.*, 1992]. For example, Chalmers trained a RAAM to auto-associate parse trees of active and passive English language sentences. After this training process had converged, the hidden layer of the RAAM yielded distributed representations for each of the sentences, thus providing the $E(\cdot)$ and $D(\cdot)$ mappings. He then trained another feed-forward network, $g(\cdot)$, to transform the resulting distributed representations of active sentences into the distributed representations of their passive counterparts. By encoding an active sentence, passing it through the feed-forward *transformation network*, and then decoding it, the system generated the corresponding passive sentence.

Because of the opacity of the representations used during holistic inference, the functions $E(\cdot)$, $D(\cdot)$, and $g(\cdot)$ must all be learned through training. To date, the representations (ie. $E(\cdot)$ and $D(\cdot)$) have been learned first, as shown in Figure 2(b). The transformational mapping $g(\cdot)$ is learned only after the representations have been completely determined. A particularly important observation is that the complexity of the mapping required for $g(\cdot)$ will depend upon the particular mappings obtained for $E(\cdot)$ and $D(\cdot)$. The feasibility of the transformational holistic approach hinges upon the learning complexity of $g(\cdot)$. When $E(\cdot)$ and $D(\cdot)$ are learned independently of $g(\cdot)$, as they were in the experiments of [Chalmers, 1990] and [Blank *et al.*, 1992], the target inference task has no effect upon the resulting representation scheme. In such a case, a low learning complexity for $g(\cdot)$ can only be expected when the ideal representations for auto-association happen to be appropriate for the given inference task.

3.2 Confluent Inference

One possible way to overcome the above limitation is to take account of the eventual inference tasks *while learning representations* ([Miikkulainen and Dyer, 1988], [Miikkulainen and Dyer, 1989], [St. John and McClelland, 1990]). The resulting representation scheme should strike a compromise between the ease of the desired inferences and the necessary auto-associative capabilities. This is the motivation behind the *confluent* approach.

Confluent inference should be viewed primarily as a representation forming mechanism. Confluence causes the distinctions that are important for the given inference task to become readily accessible within the microfeatures of the distributed representation. These distinctions are not necessarily interpretable to a human examining the representations, but they emerge so that the given inference task can be performed easily. Although confluence should be viewed as a mechanism for tailoring representations, confluent inference can be used to perform the entire inference task by itself. The FGREP algorithm [Miikkulainen and Dyer, 1988] can also be viewed in this way. In order to obtain a better understanding of the technique, the discussion and experiments focus upon the use of pure confluent inference. In a later section, the possibilities for hybrid configurations are considered, where confluence is used to shape the representations, and transformational inference is synergistically employed for the inference task.

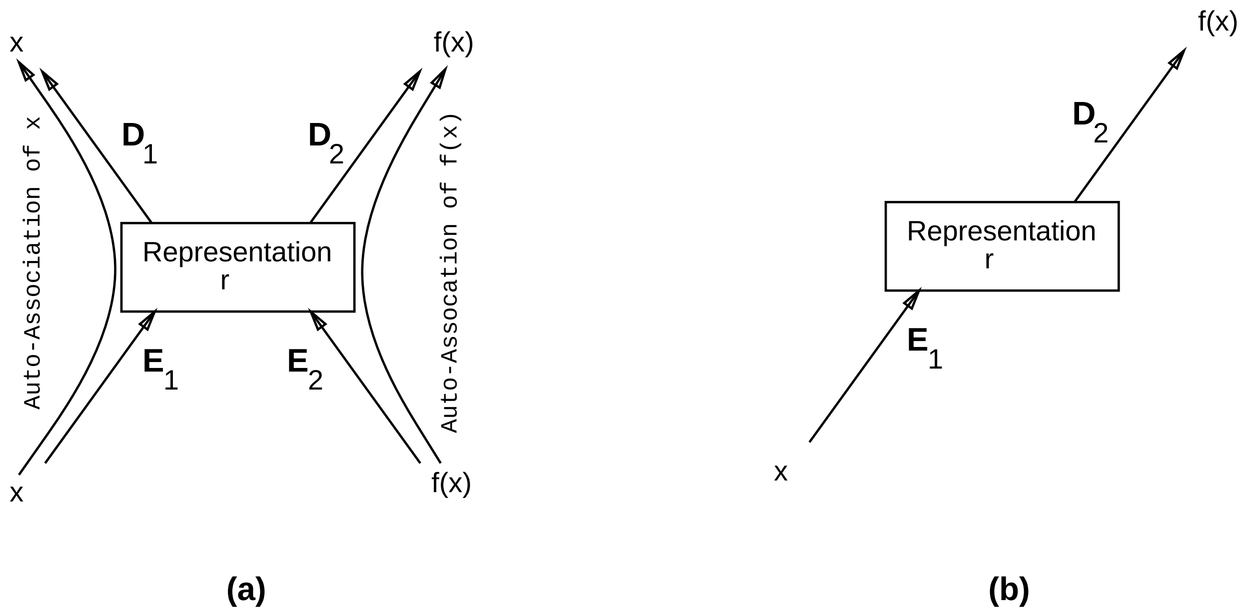

Figure 3: (a) Two way auto-association. (b) The computation of f(x) by confluent inference.

If x is an input to an inference task, and $f(x)$ is the desired output, then *confluent inference* attempts to "bring together" the representation of x with the representation of $f(x)$. The expectation is that when x and $f(x)$ are closely associated, then the inference distance between the representations for x and $f(x)$ should be small, and the transformational complexity should be low.

Taken to an extreme, confluent inference suggests that the representation for a problem, x, and its answer, $f(x)$, should have *identical* representations! The key insight is that a given representation may have two different interpretations (ie. decodings), one corresponding to the initial problem, the other to the answer of the inference task. For the English$\leftrightarrow$Spanish task considered later, the representation may be considered to be analogous to idea of *interlingua* used by the Machine Translation community. One decoder maps the interlingua into English and a different decoder maps it into Spanish. The relationships between confluent and interlingual representations are discussed later in the paper.

The inferences under consideration will be initially limited to 1-to-1 functions. In a later section, the case of general N-to-1 functions will be considered. When confluent inference is used to learn a 1-to-1 function, $f(\cdot)$, the added benefit is obtained that the inverse mapping, $f^{-1}(\cdot)$, can be acquired simultaneously.

To operationalize the confluence technique, additional encoding and decoding processes are employed. As a result, the interpretation of a particular representation depends upon which encoding or decoding is used. Let R represent the set of all possible (distributed) representations, and let X and Y be the domain and range of $f(\cdot)$ respectively. Let $E_1 : X \rightarrow R$ and $D_1 : R \rightarrow X$ be an encoding and decoding pair which is used for representing the input x to the inference task, and let $E_2 : Y \rightarrow R$ and $D_2 : R \rightarrow Y$ be a second pair used for representing the output $f(x)$ within the same space of representations. We say that the network *auto-associates* x and $f(x)$ when $D_1(E_1(x)) = x$ and $D_2(E_2(f(x))) = f(x)$ as shown in Figure 3(a). One can consider this network as being capable of representing the problem and its answer. When it is also the case that $D_2(E_1(x)) = f(x)$ as in Figure 3(b), then $f(x)$ is said to be computed by *confluent inference*.

Although $D_1(\cdot)$ and $E_2(\cdot)$ are not activated during the computation of $f(x)$ in Figure 3(b), they nonetheless play a critical role during the process of learning representations. The key point to keep in mind is that both x and $f(x)$ are combinatorial data structures with embedded constituent structure. The auto-associative pathways are necessary for learning how to build up representations of whole structures from the constituent parts by using RAAM-like training methods. Since $E_2(\cdot)$ and $D_1(\cdot)$ must be trained anyway, they provide a convenient method for obtaining the inverse function $f^{-1}(y)$ as $D_1(E_2(y))$, where $y = f(x)$.

4 The Dual-Ported RAAM

The dual-ported RAAM architecture of Figure 4 was developed and implemented in order to conduct experiments with confluent inference. The experiments conducted thus far have been design to test the abilities of pure confluent inference without a hybrid transformational component.

Given a data structure x to encode, $Encoder_1$ of Figure 4 is used to compute the representation $E_1(x)$ using the same encoding procedure as for the basic RAAM architecture. For a list, the first element is placed on the leftmost L_1 units of $Encoder_1$'s input and an *empty* token on the rightmost K units in order to obtain (on the hidden units) a representation for the one element list. This representation is then copied to the rightmost K units of $Encoder_1$'s input, the second list element placed on the leftmost L_1 units, and the process is repeated until all elements of the list have been encoded. Similarly, a representation for $f(x)$ is obtained by using this same procedure with $Encoder_2$. The basic (multiple-cycle) RAAM decoding process is used to convert a distributed representation into data structures representing x and $f(x)$ by using $Decoder_1$ and $Decoder_2$ respectively. Note that the implementation of the encoding and decoding functions $E_i(\cdot)$ and $D_i(\cdot)$ involve multiple execution cycles.

The back-propagation procedure is employed to train the dual-ported RAAM. As with the basic RAAM, the encoder-decoder pairs must be trained to auto-associate all lists and sublists, but in addition, the constraint of confluence association requires the resulting representation r to decode as $D_1(r) = x$ and $D_2(r) = f(x)$. In order to specify this process, it is necessary to distinguish between single encoding or decoding steps and the net result of encoding or decoding. Let $r_1^i = Encode_1(\langle a, r_1^{i-1}\rangle)$ represent the activations that appear on the hidden units in Figure 4 when $\langle a, r_1^{i-1}\rangle$ is placed on $Encoder_1$'s input. Let $Decode(r_j) = \langle b, r_k\rangle$ be the activations on the output units when r_j is placed on the hidden units. Similar notation is used for $Encode_2(\cdot)$ and $Decode_2(\cdot)$. Consider here a function $f(x)$ that accepts a list $x = (x_1, x_2, ..., x_n)$ as input and produces a list $f(x) = (f_1, f_2, ..., f_m)$ as output. One epoch of the training process is given in Figure 5. The algorithm assumes a 1-to-1 function so that the network is trained to produce $D_1(E_2(f(x))) = x$ as well as $D_2(E_1(x)) = f(x)$. The critical point to note is that auto-association is required for all sublists (as with the basic RAAM), but confluent association is only required for the complete list.

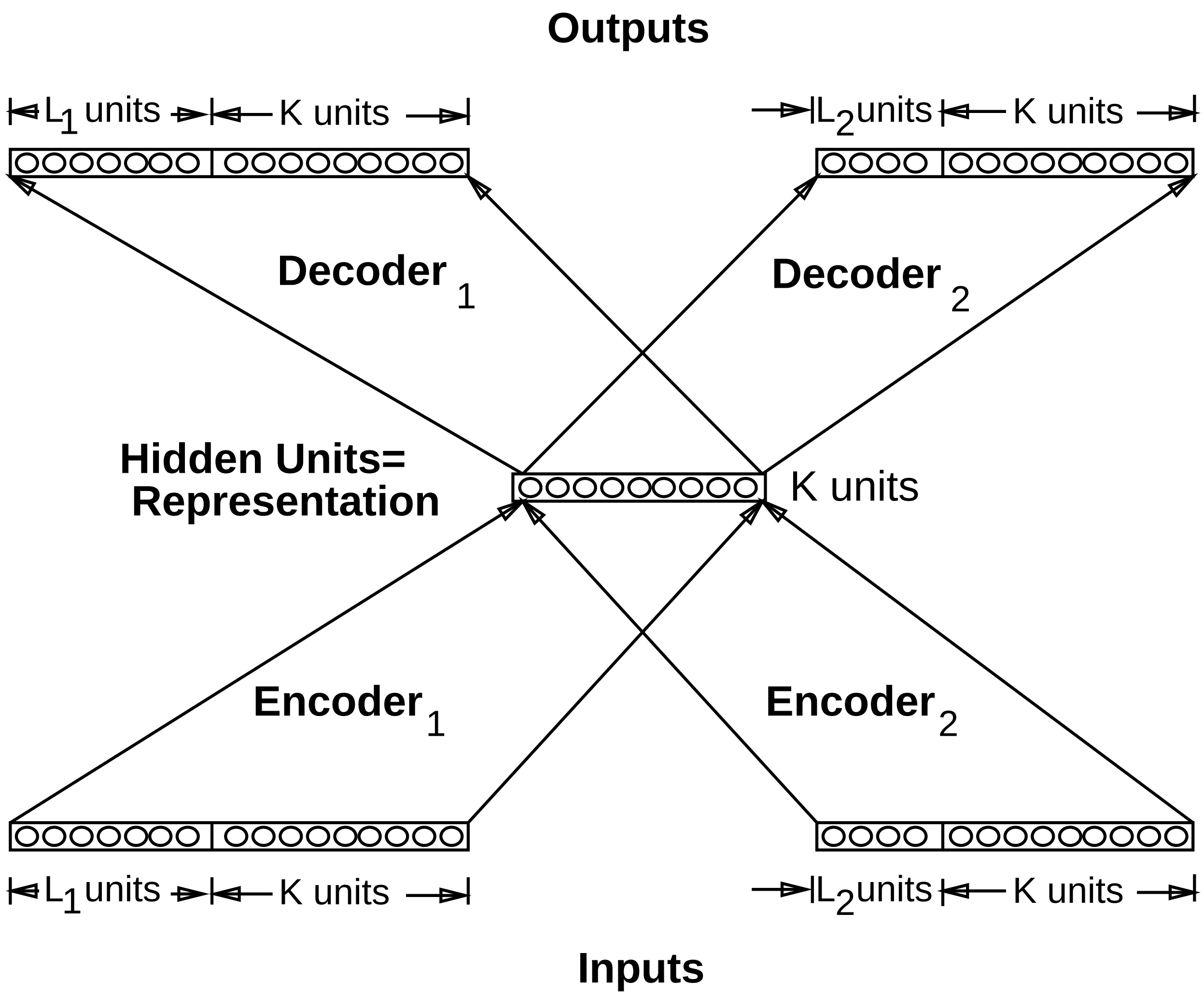

Figure 4: The Dual-Ported RAAM Architecture.

Given: $x = (x_1, x_2, ..., x_n)$, $f(x) = (f_1, f_2, ..., f_m)$

1. `let` $r_1^0 = r_2^0 = empty$.
2. `for` $i = 1..(n-1)$ `do` ;; *Auto-association*
   (a) put $\langle x_i, r_1^{i-1} \rangle$ on input to $Encoder_1$.
   (b) propagate activations through $Encoder_1$ to obtain the activations on the hidden units. Denote this as r_1^i.
   (c) propagate activations from the hidden units through $Decoder_1$ to the outputs.
   (d) invoke back-propagation on the three-layer $Encoder_1$–$Decoder_1$ network using $\langle x_i, r_1^{i-1} \rangle$ as the ideal output.
3. Repeat step 2 on the $Encoder_2$–$Decoder_2$ networks for $i = 1, ..., (m-1)$ with representations r_2^i.
4. Enforce the confluence association of x as follows:
   (a) put $\langle x_n, r_1^{n-1} \rangle$ on the input of $Encoder_1$.
   (b) propagate activations through $Encoder_1$ to hidden units to obtain r_1^n.
   (c) propagate activations from the hidden units through both $Decoder_1$ and $Decoder_2$.
   (d) invoke back-propagation using $Decoder_1 \cup Decoder_2$ as the output layer, $Encoder_1$ as the input layer, and $\langle x_n, r_1^{n-1} \rangle \oplus \langle f_m, r_2^{m-1} \rangle$ as the target output.
5. Enforce the confluence association of $f(x)$ as follows:
   (a) put $\langle f_m, r_2^{m-1} \rangle$ on the input of $Encoder_2$.
   (b) propagate activations through $Encoder_2$ to hidden units to obtain r_2^m.
   (c) propagate activations from the hidden units through both $Decoder_1$ and $Decoder_2$.
   (d) invoke back-propagation using $Decoder_1 \cup Decoder_2$ as the output layer, $Encoder_2$ as the input layer, and $\langle x_m, r_1^{n-1} \rangle \oplus \langle f_m, r_2^{m-1} \rangle$ as the target output.
6. Repeat all steps above for each training pair x,$f(x)$.

Figure 5: The Dual-Ported RAAM Training Epoch.

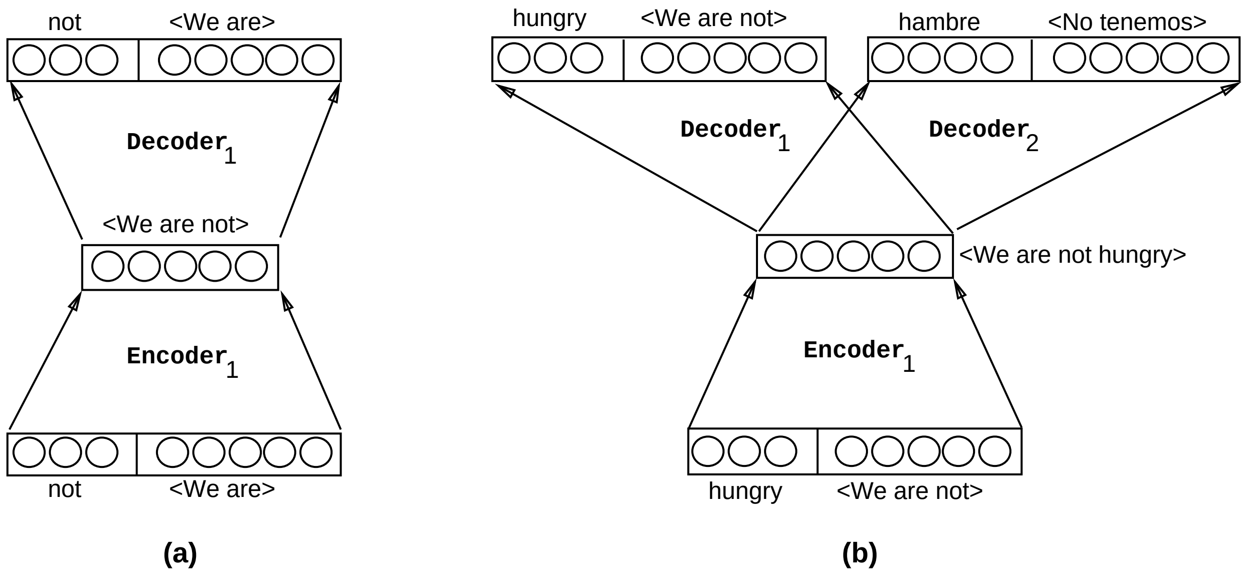

Figure 6: Training of the Dual-Ported RAAM. (a) The basic sequential RAAM configuration is used to auto-associate all incomplete sub-sentences. (b) For the completed sentence, the system is required to produce both the auto-associated output *as well as* the translation (or in general, the answer to the desired inference). Back-propagation is used at each step with the appropriate network configuration.

The training process is shown pictorially in Figure 6 for the English to Spanish translation task. For a step in the RAAM encoding process that encodes an uncompleted sentence, the normal RAAM configuration shown in 6(a) is used. The words marked by the $<$ and $>$ symbols signify that the corresponding distributed representation for that sub-sentence appears or is inserted in the designated location. Confluence is introduced by modifying the final step of the encoding process as shown in Figure 6(b). The final step requires the complete encoding to decode into both English and Spanish through $Decoder_1$ and $Decoder_2$ respectively. In the figure, the encoding for "< No tenemos >" must be obtained by using $Encoder_2$ just prior to performing this final step. Although the system only makes the problem–answer association at the final encoding step, over multiple epochs the "moving target" learning of the RAAM impacts the representations chosen for the encodings of earlier sentence fragments.

This process of associating problems with their answers only on the final step (similarly to the "*classification* paradigm" used by [Pollack, 1991]) can be contrasted with the *predictive* paradigm used by [Servan-Schreiber *et al.*, 1988], [Elman, 1990a], [St. John and McClelland, 1990], [Miikkulainen and Dyer, 1989], [Miikkulainen and Dyer, 1990], and [Lee *et al.*, 1990]. In the predictive counterpart, a configuration similar to Figure 6(b) would consistently be employed for each sub-sentence. Upon seeing the first word of the sentence ("We"), back-propagation would use the complete answer ("hambre" — "<No tenemos>") as the target output for on the second decoding pathway, while $Decoder_1$ would continue to be used as an auto-associative pathway.

Although the training process has been described in terms of lists, the dual-ported RAAM is equally applicable to any structures that can be encoded by a RAAM. For a fixed-valence tree, step 2 is applied to all non-root nodes, and step 4 is applied to the root node.

5 Natural Language Translation Task

To test the feasibility of pure confluent inference, a dual-ported RAAM was implemented and applied to a small English $\leftrightarrow$ Spanish domain. Allen [1987] also previously applied a back-propagation network to a small English to Spanish translation task using a multi-layer feed-forward network. Unlike the structure-sensitive encodings being studied here, Allen restricted the maximum sentence length so that all words could be simultaneously applied at the input and output layers. The translation task provides an interesting domain for experimenting with structure-sensitive holistic inference. It is a domain where pure transformational holistic inference would be expected to perform poorly since the regularities and vocabulary in each language are distinct. When both English and Spanish sentences are auto-associated by a pure RAAM, this distinctness causes the RAAM to develop unrelated encoding schemes for each language, and the holistic translation transformation is very complex. In fact, [Allen, 1987] reports also encountering this same phenomena. He trained two feed-forward auto-associating networks to develop hidden-unit representation of sentences in each language, and then trained a network to transform from the English net's hidden representation to the Spanish net's representation. His "preliminary" results indicated that "apparently the types of features extracted in the two auto-associator networks are not easily coordinated." However, by using confluent inference, the dual-ported RAAM develops closely related encoding schemes for the two languages.

A corpus of 216 possible English-Spanish sentence pairs (ie. 432 total sentences) were enumerated from a vocabulary of 36 english and 36 spanish words. The words and their (localist) encodings are shown in Figures 7 and 8. The encoding scheme was quickly chosen, based only upon a subjective feel for the style of representation used by [Pollack, 1990] and [Chalmers, 1990], with no additional time expended on any clever engineering of the patterns. The first group of bits correspond to word category (ie. verb, subject-noun, pronoun, preposition, adverb, determiner, adjective, simple-noun, or place). The middle group of bits correspond to a rough intuitive notion of plurality. For verbs, this corresponds to categories such as first-person-singular or third-person-plural, for most nouns they encode simple plurality, and for other words they are set arbitrarily. Finally, the last group selects the specific identity of the word. A number of interesting surface phenomena occur in these sentence pairs making the translation task non-trivial. The number of words and the word ordering commonly differ.

- He has it. $\leftrightarrow$ Lo tiene. ("It he_has")
- We do not want it. $\leftrightarrow$ No lo queremos. ("Not it we_want")

There are distinctions made in each language not made by the other. For example, in the following sentences, the english word "is" maps to three different verbs in Spanish.

- He is a student. $\leftrightarrow$ Es estudiante.

	Category									Plurality			Identity									
do	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0
does	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
want	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
wants	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
have	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0
has	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
am	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
is	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
are	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
I	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
you	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
he	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
Reid	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
they	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
we	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
it	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
from	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
not	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
happy	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
angry	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
fine	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
here	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
young	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
old	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
right	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
sleepy	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
hungry	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
thirsty	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
professor	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0
professors	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0
student	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0
students	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
money	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	1	0	0
Pittsburgh	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
California	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 7: The English Word Encodings.

	Category									Plurality				Identity					
quiero	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
quiere	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
quieren	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
queremos	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
tengo	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
tiene	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
tienen	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
tenemos	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
estoy	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
está	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
están	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
estamos	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
soy	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
es	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
son	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
somos	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Usted	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
Reid	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
lo	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
no	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
contento	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0
contentos	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0
furioso	0	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0
furiosos	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	0	0
bien	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0	1	0	0	0
aquí	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0	0	1	0	0
joven	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0
jovenes	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	1	0
viejo	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1
viejos	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1
profesor	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0
profesores	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0
estudiante	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0
estudiantes	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0
Pittsburgh	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0
California	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	1	0	1	0
razón	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	0	0
sueno	0	0	0	0	0	0	1	0	1	1	1	1	1	0	1	0	0	0	0
hambre	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	1	0	0	0
sed	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	1	0	0
dinero	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	0	1	0

Figure 8: The Spanish Word Encodings.

- He is happy. $\leftrightarrow$ Está contento.
- He is hungry. $\leftrightarrow$ Tiene hambre.

Similarly, the spanish verb "tener" can map to different english verbs:

- Tienen razón. $\leftrightarrow$ They are right.
- Tienen dinero. $\leftrightarrow$ They have money.

The verb conjugations between the two languages are not identical. For example, in Spanish the following conjugations are the same while in English the conjugations differ ("are" vs. "is"):

- You are young. $\leftrightarrow$ Usted es joven.
- Reid is young. $\leftrightarrow$ Reid es joven.

Also, different spanish conjugations exist for the english conjugation "are":

- You are here. $\leftrightarrow$ Usted está aquí.
- We are here. $\leftrightarrow$ Estamos aquí.
- They are here. $\leftrightarrow$ Están aquí.

These surface phenomena make the task particularly ill-suited for trivial word-for-word based translations. Since the set of sentences seem to require a semantic association rather than a purely syntactic one, the emergent representational micro-structure should reflect information beyond simple compositional structure.

The first experiment was designed to test memorization capabilities and measure the extent to which confluence is achieved, with no consideration of generalization proficiency over unseen sentences. All 216 sentence pairs were used for training. The number of hidden units was $K = 40$, and $L_1 = 22$, $L_2 = 19$ yielding a network topology[2] of $(62 \oplus 59)$–40–$(62 \oplus 59)$. The learning rate r began at $r = 0.1$, but was decreased to $r = 0.01$ near the end of training. The momentum m began at $m = 0.3$ but was quickly increased to $m = 0.9$, and eventually increased to $m = 0.97$ near the end of training. A terminal tolerance of $\tau = 0.2$ and a non-terminal tolerance of $\nu = 0.05$ were used. After 5200 epochs all 432 sentences and sentence translations successfully memorized, with only 9 words being only "weakly" learned where at at least one bit in the word had an activation between 0.2 and 0.8.

If confluence is taking place during training, then the internal distributed representations for equivalent Spanish and English sentences should be very closely related. An examination of the resulting representations verified that this is very much the case. The resulting distributed representations for a small sampling of the sentences is shown in Figure 9. For the entire corpus of sentences, it is quite clear that the representational confluence is very pronounced.

[2]As Figure 4 shows, the topology is similar to having two different networks with topologies of 62–40–62 and 59–40–59 which share the hidden same hidden units. The $\oplus$ notation summarizes this.

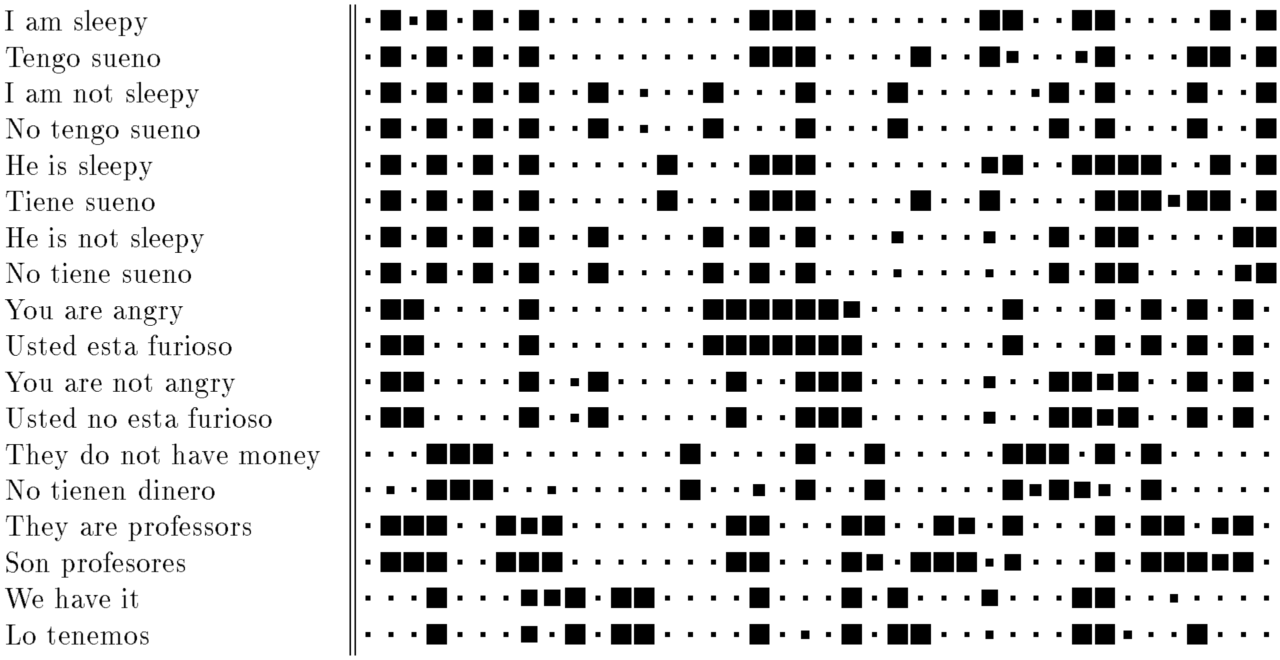

Figure 9: Distributed Representations Obtained During Experiment 1.

An interesting exercise is to determine whether any hidden unit consistently responds to particular identifiable semantic or structural features [Hinton, 1986]. It seems reasonable, for example, that a unit might dedicate itself to representing sentence polarity (eg. "I have it" vs. "I do not have it."). Other units may dedicate themselves to capturing the particular subject or verb of the sentence, etc. However, as with FGREP representations ([Miikkulainen and Dyer, 1988], [Miikkulainen and Dyer, 1990]), these sorts of clear, unambiguously interpretable microfeatures did not occur. Identifiable microfeatures seem to occur only mildly over small groups of closely related sentences, but not at all consistently over the entire training set. For example, the sixth unit from the left consistently encoded whether the sentence was negative when the verb of the sentence was "to want" (querer) or "to have" (tener), but did not correlate at all with sentence polarity in any of the other sentences. Many other units were even more opaque. The sentence structure and content really do appear to be truly distributed within the representation.

The second experiment tested the generalization capabilities of the entire process. The sentence pairs were randomly partitioned into two equally sized groups, and one group (of 108 sentence pairs) was used for training. Using the same training parameters as in the first experiment, convergence was reached in 5000 epochs. As expected, 100% of the sentences and translations used for training were memorized correctly. The remaining 108 sentence pairs (216 sentences) were then used for testing the generalization accuracy of the system. The generalization accuracy is divided into two parts, the auto-associative accuracy (ie. the ability to encode and decode a sentence to obtain the original sentence) and the translational accuracy (ie. the ability to correctly translate a sentence into the other language). The trained network correctly auto-associated 49% (105/216) and correctly translated 33% (72/216) of the testing sentences. For these figures, only the sentences that were reproduced exactly were counted as correct.

A common error on the testing sentences (for both auto-association and translation) resulted from the premature truncation of a sentence. This was a result of the particular scheme used to detect list termination. Specifically, the decoding process was terminated whenever any output unit in the left portion of the output differed from the closest matching word by more than 0.2[3]. This problem is a result of the RAAM architecture, but not a result of the addition of the dual port or of confluent inference. Furthermore, there are many obvious techniques one could employ to alleviate the problem, and it seems reasonable to assume that with some additional effort, the problem would cease to be significant[4]. Therefore, it is more enlightening to examine the generalization accuracy while assuming that the list termination problem is solved. This is easy to do because one can simply count the number of words in the target testing sentence and use that value for deciding how many decoding cycles to perform. This causes some words to be output even though they may have units with activations between 0.2 and 0.8. In these cases the closest match in the dictionary is output. This alteration isolates the inference process from the list termination problem. When the testing sentences were printed out in this fashion, the resulting generalization accuracies were 85% (183/216) for auto-association and 70% (151/216) for translation. Another 94% (31/33) and 82% (53/65) of the incorrect sentences were "almost" correct differing either by a single erroneous word or an incorrect subject with a consistent verb conjugation.

While 30% of the testing sentences were translated incorrectly in the previous experiment, half this amount (15%) were unsuccessfully auto-associated. This suggests that a considerable hindrance to the translational accuracy is *not* confluent inference, but rather the ability of the RAAM to auto-associate (ie. represent) the sentences. Chalmers [Chalmers, 1990] also found that errors due to the RAAM's mistakes in generalizing its representations to unseen sentences dominated the accuracy of his experiments. To isolate the two phenomena, he trained to a net auto-associate all possible sentences. An analogous experiment follows.

The third experiment was designed to test the generalization capabilities of confluent inference while factoring out the effects of incorrect auto-associative generalization by the RAAM. For this experiment, the confluent training process of Figure 5 was used for the first 108 sentence pairs. Pollack's standard RAAM training scheme was applied to the other 108 sentence pairs by independently using the english sentence to train the $Encoder_1$–$Decoder_1$ pair and the spanish sentence to train the $Encoder_2$–$Decoder_2$ pair. After 4200 training epochs (using the same learning parameters as the previous experiments), the network had memorized all 432 auto-associations as well as the 216 translations from the training set[5]. The remaining 108 sentence pairs were then used to test the generalization accuracy of

[3]It is not particularly surprising that this would be a common error since any time any word within a sentence is generated weakly, the entire sentence is truncated and counted as incorrect.

[4]An end of sentence mark (ie. a period or *stop* mark) has been used by [Miikkulainen and Dyer, 1990] and [Blank *et al.*, 1992]. [Miikkulainen and Dyer, 1990] report that "the system learns to output the period quite early in the training," thus suggesting that the sentence termination problem is probably rather easy to overcome.

[5]All but five words were within $\tau = 0.2$. The remaining five words were within $\tau = 0.3$.

Testing Sentence	Erroneous Translation	Correct Answer
I do not want it	No lo *quieren* *	No lo quiero
No lo quiero	They *do* not want it *	I do not want it
I have it	Lo tenemos *	Lo tengo
I do not have it	No lo *tienen*	No lo tengo
No lo tengo	They *do not want* it *	I do not have it
I am not a professor	No *somos profesor*	No soy profesor
I am not a student	No *somos* estudiante	No soy estudiante
No lo tiene	He does not *want* it *	He does not have it
Es de California	*We is* from California	He is from California
Usted no lo quiere	*He* does *not* want it *	You do not want it
Usted no lo tiene	Reid does not have it *	You do not have it
You are a professor	*Es* es professor	Usted es profesor
Usted es profesor	He *wants a professor*	You are a professor
Usted es estudiante	He *wants a* student	You are a student
You are not a student	Usted no es *profesor* *	Usted no es estudiante
You are not from California	*Estudiante* no es de *sueno*	Usted no es de California
Usted no es de California	He is not *from* California *	You are not from California
You are not from Pittsburgh	*Estudiante* no es de Pittsburgh	Usted no es de Pittsburgh
Usted no es de Pittsburgh	He is not *from* Pittsburgh	You are not from Pittsburgh
They want money	Queremos dinero	Quieren dinero
They have money	Tenemos *dinero*	Tienen dinero
No tenemos dinero	*He* have *not* have *money*	We do not have money
We are not professors	*Lo* somos profesores	No somos profesores

- Words in italics were only weakly activated with at least one output unit between 0.2 and 0.8.
- Responses marked with asterisks appeared in the training set.

Figure 10: Incorrectly Generalized Translations in Third Experiment.

translation. The system translated 89% (193/216 = 96/108 English → Spanish and 97/108 Spanish → English) of the sentences perfectly. Of the mistakes, 91% (21/23) could be considered near misses, differing by only one incorrect word or by an incorrect subject with the verb agreeing in conjugation. All of these sentences can properly be auto-associated by the system; therefore, these numbers reflect the true accuracy of the confluent inference process for this translation task. Since associational abilities give holistic inference its intriguing potential, the mistakes made by holistic inference engines are sometimes more interesting than the success rates obtained. Figure 10 shows all testing sentences that were incorrectly translated by the network in this third experiment.

6 Variations

The presentation of confluent inference thus far has only considered the case of a single transformation-like invertible inference task. A number of variations are also possible and are considered in this section.

6.1 N-to-1 mappings

The English↔Spanish translation task in the previous section was an example of a 1-to-1 mapping. When inference tasks are 1-to-1, confluent inference can be used to obtain both the forward and inverse mappings. These mappings are a special case, but the application of confluent inference is not limited to solely invertible mappings. The generalization to N-to-1 mappings is quite simple and is presented here. Obviously, by going to N-to-1 mappings, one must give up the luxury of automatically obtaining the inverse.

The training process for a dual-ported RAAM attempts to achieve four constraints for each training instance: $D_1(E_1(x)) = x$, $D_2(E_2(x)) = x$, $D_2(E_1(x)) = f(x)$, and $D_1(E_2(f(x))) = x$. These correspond to steps 2, 3, 4, and 5 of the training algorithm in Figure 5. Extending this training procedure to N-to-1 functions requires nothing more than simply deleting the last constraint. The only minor subtlety that exists when step 5 is deleted is that the loop in step 3 must be extended for $i = 1, ..., m$. This is because originally step 3 handled the auto-association of all sub-sentences, but the final auto-association of the entire sentence occurred as part of step 5.

While technically N-to-1 functions present no problem for confluent inference, our basic intuitions behind confluence must change slightly. Consider two different inputs which must produce the same output (ie. $f(x_1) = f(x_2)$). It is no longer ideal for confluent inference to map a problem and its answer to identical representations. Since the system must still maintain its auto-associative capabilities, the representation for an input must encode more information than the representation of the corresponding output. In other words, in order to maintain auto-associative abilities, the system must be able to distinguish between x_1 and x_2 from their representations; however, since $f(x_1) = f(x_2)$, it is not possible to make such a distinction from the representation of $f(x)$ alone. In the case of an N-to-1 function, confluence intuitively suggests that some semantic *subset* of a problem and its corresponding answer should "come together."

6.2 Detectors and Parallel Decoders

Up to this point, we have considered the case where both the inputs and outputs of an inference task possess constituent structure. [Blank *et al.*, 1992] identifies three types of holistic operations (not including confluent inference): decoders, detectors, and transformers. Decoders and detectors are distinguished in that the output does not contain constituent structure. The lack of constituent structure in the output means that the complicated RAAM-like encoding process for the output representations is unnecessary. Thus, there is no distributed representation of the answer. For the dual-ported RAAM architecture in Figure 4, the second encoder-decoder pair can be replaced with a simple feed-forward

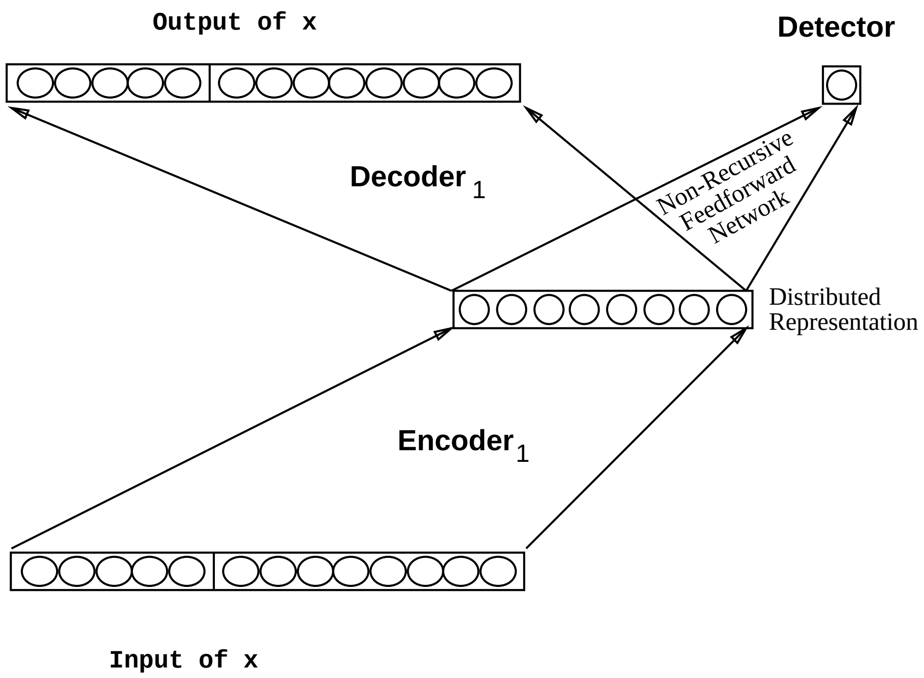

Figure 11: Using a Detector (or Parallel Decoder) with Confluent Inference.

decoder or detector network.

A *parallel decoder (PD)* extracts a particular constituent from the distributed representation of a sentence. The term "parallel decoder" is borrowed from [Blank *et al.*, 1992] and should not be confused the decoder half of an encoder-decoder pair. Simple PDs are somewhat interesting since they perform distal access [Newell, 1980] in a single step for instances where pointer following may require multiple steps. For example, a simple PD might return the last element of an encoded list.

While a simple PD extracts information that lies at a fixed location within a data structure, more complicated context-dependent PDs may also be envisioned. Consider, for example, a system that accepts as input a sentence as a sequence of words, and constructs a distributed representation (a "Sentence Gestalt") for the sentence. A complicated PD may then be used to extract the `agent` of the sentence, even though the precise location of the agent within the sentence is not fixed. The system of [St. John and McClelland, 1990] can thus be viewed as a very interesting instance of a complex structure-sensitive, holistic PD.

A *detector* is used to holistically determine whether a simple proposition holds within an encoded data structure. For example, an "AGGRESSIVE-ANIMAL" detector might return `TRUE` just in case a sentence contains a reference to an aggressive animal. A more complicated example might be a *reflexive detector* that returns true just in case the subject and object of an encoded sentence are the same [Blank *et al.*, 1992]. Not surprisingly, confluent inference can also be used within the context of detectors and PDs to help shape the representations to match the inference task. Because the final result of the inference process does *not* possess constituent structure, the total process is somewhat simplified. Figure 11 shows the resulting architecture. As with the confluent training procedure (Figures 5 and 6), all sub-structures

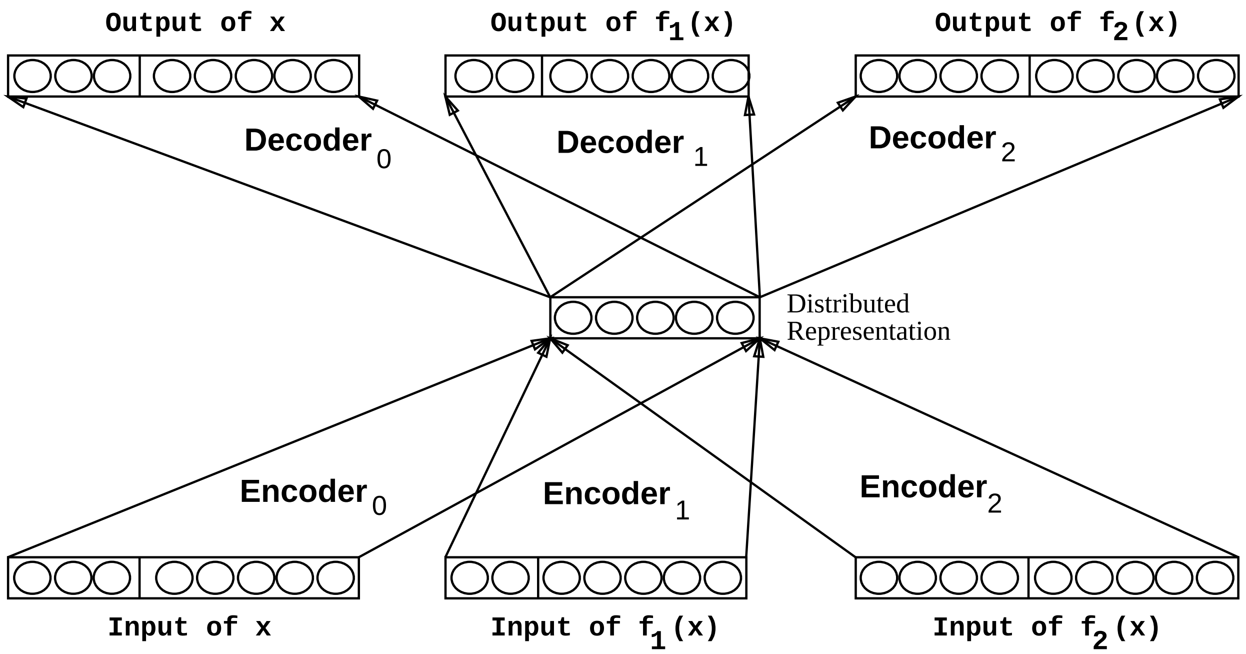

Figure 12: A Multi-ported RAAM architecture.

of the input are auto-associated, then during the final encoding of the complete structure, the hidden units are forced to produce both the auto-associated output and the final detector or PD output.

The use of confluence in the context of detectors and PDs is very closely related to the SG model [St. John and McClelland, 1990] and thus it aids in comparing that with the current work. The two differences are that the SG model is "predictive," confluent inference is not, and that the SG model does *not* require the ability to auto-associate the input. Thus, their Sentence Gestalt is not required to preserve information that is not used within the inference task.

6.3 Multiple inference tasks

When a system has multiple structure-sensitive holistic inference tasks, the distributed representations should be formed so as to make all inference tasks as easy as possible [Hinton, 1986]. Up to this point, only a single inference task has been considered, but the techniques extend straightforwardly to the case of multiple inference tasks. The generalization to M inference tasks simply requires the use of $M+1$ encoder-decoder pairs, where the first pair corresponds to the problem input and the remaining M pairs correspond to each additional inference task. A multi-ported RAAM architecture is diagramed in Figure 12.

FGREP has also been used to form representations that are appropriate for multiple inference tasks [Miikkulainen and Dyer, 1989]. An important difference between FGREP and confluent inference is pronounced in this case. When a particular distinction is never used by any inference task, the FGREP method will eliminate that distinction from the representation (as does Hinton's [1986] family tree system). For example, because the words `man`, `woman`, `boy`, and girl are always used in the same way in [Miikkulainen and Dyer, 1990], the representations for each of these words becomes identical. For this reason, [Lee *et al.*, 1990] argues for the development of task-independent representations, claiming that because their Distributed Semantic Representations (DSRs) are learned independent of any particular task, they are *portable* to tasks outside of the training environment.

Confluent inference stands in strong contention with the notion that useful DSRs should be learned independent of their target inference tasks. While task-independent representations *may* preserve all relevant information, if that information is not nicely reflected within the micro-structure of representation, then holistic inference will be difficult, generalization will be poor, and the system may have to instead settle for "rational" inference [Hinton, 1990] by decomposing a representation into its constituent structure before performing inference. Since the DSRs of [Lee *et al.*, 1990] are formed by achieving something roughly summarized as the mutual auto-association of concept and proposition encodings, one would expect those representations to be readily portable to other tasks *only* where the important features of the new task are a subset of those necessary for the mutual auto-association task.

Although confluent inference stands in opposition with [Lee *et al.*, 1990] on the idea of task-independent acquisition of representations, it is actually more similar to DSRs than to FGREP with respect to *portability*. Because confluent inference maintains enough information in its representations to complete the auto-association task, the lack of a particular distinction by the given inference tasks does *not* result in such information being thrown away in the representation. Nevertheless, more opaque encodings will usually be obtained for distinctions that are not used within any of the training tasks.

When the sample of inference tasks that are used for training are representative of other tasks that the system may later need to holistically compute, then the important distinctions will become "well-entrenched" [Goodman, 1983] and the resulting representations should be quite portable over that set of tasks.

7 Why Confluence Works

Confluent inference attempts to achieve a close association between the input and output pairs of an inference task by causing the association to be overtly reflected in the respective representations. The association appears in the form of very similar representations for a problem and its answer. Determining the appropriate scope for the confluent technique consists of understanding when the confluence of the two representations is likely to occur.

The technique attempts to summarize the input and output data structures in terms of a common set of microfeatures. Because the data structures may be an arbitrary size and shape, and the microfeatures are limited to a fixed number of units, a compression must take place. In order for this compression to be effective, the system should not simply partition the units into input microfeatures and output microfeatures; instead, units must be shared between both input and output. These units can be viewed as extracting common semantic gestalt properties from either source. Furthermore, since a certain aspect of an inference task may depend upon the "whole" of the input, rather than just an individual constituent, there is additional pressure to combine the representations in the form of common microfeatures. If this were not the case, then during encoding, those units partitioned as output representations would have to be filled in through a complicated inferential process which is not immediately related to individual constituents of the input structure.

From these considerations, we can conclude that the success of confluent influence rests upon the presence of deep semantic and/or syntactic commonalities between an input problem and its answer. Thus, when one is attempting to subjectively evaluate the appropriateness of confluent inference for a given task, it is useful to think of it in the following way. If the semantic content underlying the inputs (ie. the high-level or "deep" meaning) significantly overlaps the semantic content of their corresponding outputs, then confluence will probably be beneficial. Because a RAAM also encodes syntactic structure uniformly with semantic content, overlap in "deep' syntactic regularities contribute in a similar fashion. When significant overlap in higher-level meaning does not occur, then confluent inference will probably not contribute to the computation. When only a partial overlap occurs, then confluent inference may be useful for leveraging that overlap, but transformational inference may be more appropriate for the remainder of the task. Hybrid approaches are considered in the next section. One should also always keep in mind that holistic computation, in whatever form, may not be appropriate for many tasks (cf. "intuitive" vs. "rational" inference [Hinton, 1990]).

Consider what happens when confluent inference is applied to a task where semantic overlap does not occur. For example, suppose input-output pairs for a function are created by randomly pairing a set of data structures. Provided that the network has a weak-enough bias (eg. enough hidden units), just as a feed-forward back-propagation network can be used to memorize arbitrary mappings, the confluence training algorithm should eventually memorize the training set[6]. Since common semantic features do not exist, the associated representations will not exhibit confluence. In other words, in this extreme case, the representations for a problem and its answer will *not* "come together." Generalization abilities will likewise be poor and encoding efficiency will be low.

7.1 Interlingua

Confluent representations in the language translation experiment play the same role as *interlingua* representations in the Machine Translation community. For this particular task, there are a few minor differences. While interlingua has long been a popular idea, the design of a sufficiently powerful intermediate language has been *the* primary impediment to constructing effective interlingual translation systems [Nirenburg, 1989]. In contrast, confluent representations are not hand crafted, and the important distinctions emerge automatically. Another difference is that confluent representations routinely capture semantic as well as syntactic features, while according to [Drozdek, 1989], "the attitude implicitly present in the interlingual method takes one to another extreme, ie. to neglecting the syntax altogether and focusing entirely on semantics." Historically, the primary attraction of interlingua for translation has been the reduction in the number of translators necessary for going between n different languages. The motivations underlying confluent inference are completely unrelated to this concern. But perhaps the most blatant difference is that the confluent representation

[6] It is not clear, however, that the "moving target learning" of a recurrent system like the RAAM will always be guaranteed to converge. In practice, it always seems to.

of a given concept might not be precisely fixed. Instead, the precise representation may vary by small amounts depending upon the source language (Figure 9). These may correspond to small language-dependent connotations that arise as a result of relationships in usage with respect to other words or constructs in the language, and these cannot be easily captured in the other language. Examples of such phenomena may include certain puns, subtle ambiguities, and many other variables of language perception such as those discussed in [Levelt, 1978, pages 21–47].

For tasks other than natural language translation, the analogy to interlingua becomes more vague. As we consider variations (eg. N-to-1 mappings) of the inference task, it is hard to identify any relevant relationship between confluence and interlingua.

8 Hybrid Approaches

As discussed earlier, confluence should be viewed primarily as a representation forming mechanism. As such, it should complement transformational inference rather than replace it. As discussed in the previous section, when some aspects of the inference task are not likely to be readily summarized by high-level semantic features, then pure confluent inference may not be the most appropriate approach. In this case, it may be easier to utilize transformational inference for some parts of the inference task. In this section, two possible approaches for obtaining a synergistic combination are presented.

In a hybrid architecture, an extra network $g(\cdot)$ is introduced to perform a holistic transformation upon the distributed representations. The transformation computes the distributed representation of $f(x)$ from the distributed representation of x. Recall that for pure confluent inference, $D_2(E_1(x)) = f(x)$. When $g(\cdot)$ is inserted, then $D_2(g(E_1(x))) = f(x)$ and it is said that $f(x)$ is computed by a combination of confluent and transformational inference. Even in the hybrid case, the auto-association pathways are still maintained, such that $D_1(E_1(x)) = x$ and $D_2(E_2(f(x))) = f(x)$.

In the simplest hybrid architecture, confluent and transformational inference are decoupled. Confluent inference is applied only during the early stages of training while representations are initially being formed. During this stage, the distributed representations of a problem and its answer tend to move together, and the eventual transformation task is biased towards simplicity. At some stage, confluent inference is turned off, but by this time it will have exerted an influence over the eventual representations that will be formed by the system, and the final transformation will be simplified as a result.

When certain aspects of an inference task are ill-suited for confluent inference, the mapping from x to $f(x)$ would be expected to converge slowly during the execution of the confluence training algorithm (Figure 5). Thus, a natural point to turn off confluent inference is when the desired auto-association accuracy is achieved, independent of inferential accuracy. At this point, all necessary data structures can be represented by the system, but an additional transformation may be necessary in order to accomplish the desired inferential accuracy.

As with the dual-ported RAAM in Figure 4, while confluent inference is turned on, the representation for a problem and its answer are treated as if they share the same representation space as shown in Figure 13(a). However, after confluent inference is turned off, the two

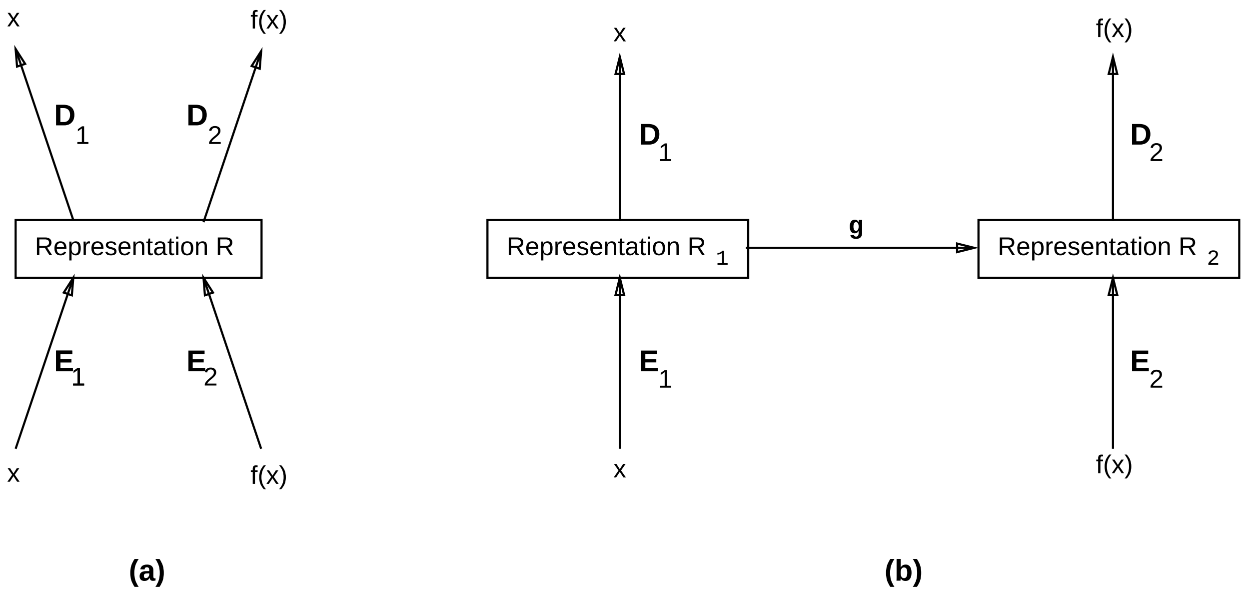

Figure 13: Decoupled Hybrid Architecture (a) While confluent inference is on, the representation spaces for a problem and its answers are shared. (b) After confluent inference is turned off, the two representation spaces are treated as if they are separate and distinct.

representation spaces are treated as distinct as diagramed in Figure 13(b). Two different encoder–decoder pairs are still used.

The simple decoupled scheme closely resembles the approach used for transformational inference by [Chalmers, 1990] and [Blank *et al.*, 1992]. The difference is that confluent inference is harnessed early on in order to influence the eventual representations.

The second approach for obtaining a hybrid architecture considers confluent and transformational components simultaneously. The confluent mapping is learned at the same time that the transformational mapping is learned, and both components are active during the entire training process.

The hybrid architecture is shown in Figure 14. The representation spaces for the problem and its answer are distinct throughout the entire process, a feature that allows a different number of units to be used in each representation space. The training process for this configuration is almost identical to the confluence training process for an N-to-1 function. The only difference is that during the confluence step (Step 4 in Figure 5), the second set of representation units are treated as an extra hidden layer during back-propagation.

It should be evident that in both hybrid approaches, the inverse of a 1-to-1 function is no longer automatically obtained as it was with pure confluent inference. To obtain the inverse, a second transformation layer must be included in the opposite direction in order to compute the inverse inference. The handling of this second transformation is straightforward in both configurations.

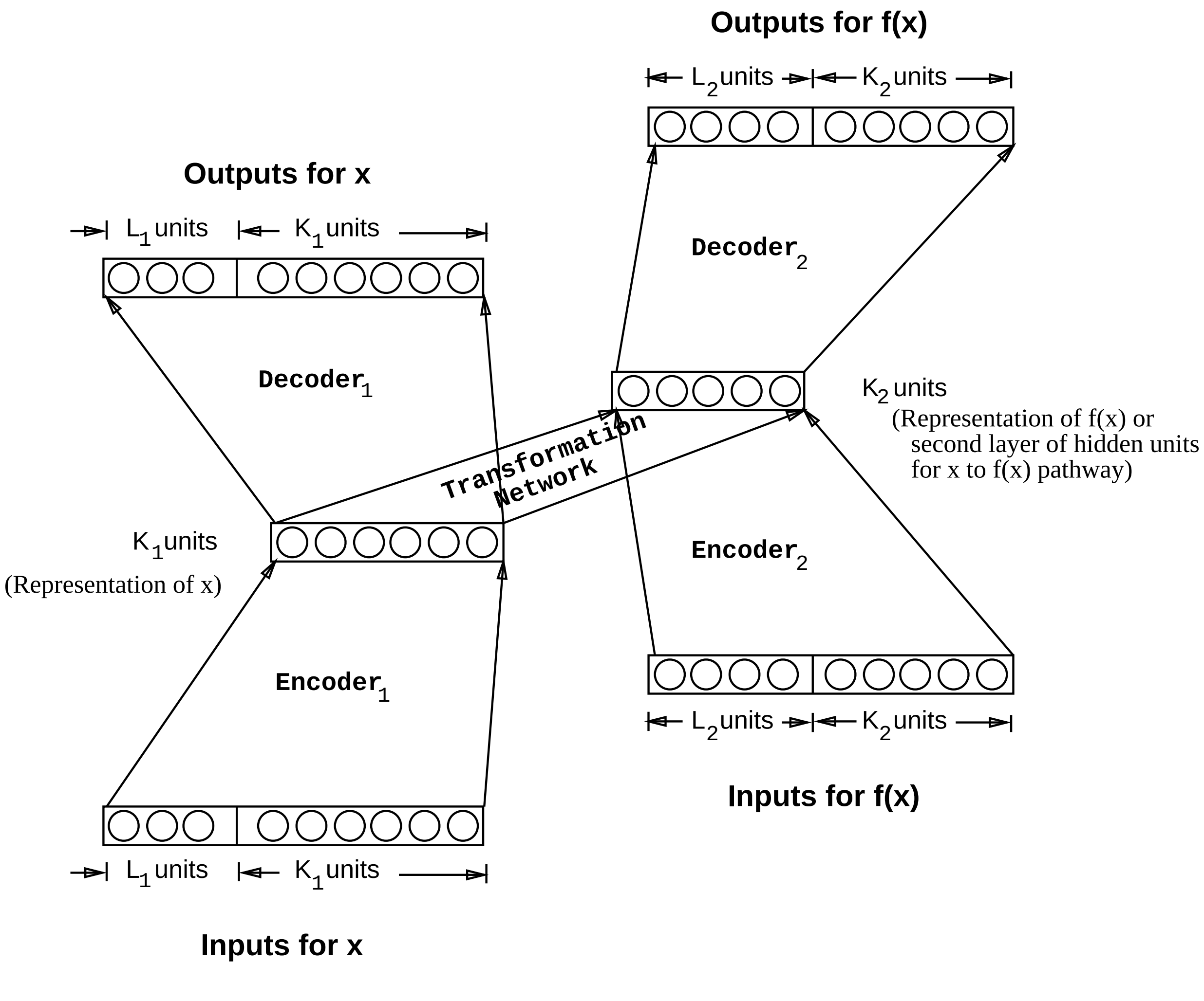

Figure 14: Coupled Hybrid Architecture.

9 Conclusion

There is widespread agreement that interesting intelligent behavior requires the maintenance and manipulation of compositionally structured data. Classically, structure-sensitive computation is performed via the explicit traversal and composition of constituent elements. However, the recent emergence of recursive connectionist representations has created the possibility for a vastly different mode of computation: *holistic inference*. By harnessing the emergent micro-structure in these distributed representations, holistic inference maps directly from the representation of a problem to the representation of its answer in a gestalt fashion, without accessing the constituent elements or relations within the data. Besides being very fast (usually constant time), there is also some hope that the associational abilities of neural networks may may result in additional benefits for employing holistic inference.

Transformational (structure-sensitive) holistic inference was introduced and successfully demonstrated by [Chalmers, 1990]. Because the representations are learned independently from the inference task, pure transformational inference will generally be infeasible unless the representations useful for auto-association are also appropriate for the inference task. A second form of holistic inference, *confluent inference*, was introduced in order to overcome this difficulty. Confluent inference accounts for the inference tasks during the formation of representations by attempting to "bring together" the representation of a problem with the representation of its answer. The intended result is that the transformational mapping (corresponding to the given inference) from problems to answers becomes as simple as possible. A dual-ported extension to Pollack's RAAM architecture [Pollack, 1990] was devised, implemented, and used to test these ideas. In a small English $\leftrightarrow$ Spanish translation task, by using pure confluent inference the system perfectly translated 89% of the testing sentences that were not in its training set. Since the task seems particularly ill-suited for a pure transformational holistic approach, the encouraging results indicate that confluence extends the feasibility of holistic approaches for structure-sensitive computation. These experiments only demonstrate the possibility of holistic techniques. The ultimate power and feasibility will rest upon further development and improvement of reduced description architectures [Hinton, 1990] and upon the harnessing of synergy between confluent and transformational methods.

Acknowledgements

I am grateful to Dave Touretzky for his helpful and influential comments on an early draft of this paper.

References

[Allen, 1987] Robert B. Allen. Several studies on natural language and back-propogation. In *International Conference on Neural Networks*, pages II–335–341, 1987.

[Blank *et al.*, 1992] Douglas S. Blank, Lisa A. Meeden, and James B. Marshall. Exploring the symbolic/subsymbolic continuum: A case study of RAAM. In J. Dinsmore, editor, *Closing the Gap: Symbolism vs. Connectionism*. Lawrence Erlbaum Associates, 1992. To Appear.

[Chalmers, 1990] David J. Chalmers. Syntactic transformations on distributed representations. *Connection Science*, 2(1 & 2):53–62, 1990.

[Drozdek, 1989] A. Drozdek. Interlingua in machine translation. In *Seventeenth Annual ACM Computer Science Conference*, page 434. ACM Press, February 1989.

[Elman, 1990a] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–212, 1990.

[Elman, 1990b] Jeffrey L. Elman. Structured representations and connectionist models. In Gerald Altmann, editor, *Computational and Psycholinguistic Approaches to Speech Processing*. Academic Press, New York, 1990.

[Fodor and Pylyshyn, 1988] J. A. Fodor and Z. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71, 1988.

[Goodman, 1983] Nelson Goodman. *Fact, Fiction, and Forecast*, chapter Chapter III (1953). Harvard University Press, fourth edition edition, 1983.

[Hinton *et al.*, 1986] G.E. Hinton, J.L. McClelland, and D.E. Rumelhart. Distributed representations. In D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* **1***: Foundations*, chapter 3, pages 77–109. MIT Press, Cambridge, MA, 1986.

[Hinton, 1986] G.E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Cognitive Science Society Conference*, pages 48–54, Hillsdale, NJ, 1986. Erlbaum.

[Hinton, 1990] Geoffrey E. Hinton. Mapping part–whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–75, 1990.

[Lee *et al.*, 1990] Geunbae Lee, Margot Flowers, and Michael G. Dyer. Learning distributed representations of conceptual knowledge and their application to script-based story processing. *Connection Science*, 2(4):313–345, 1990.

[Levelt, 1978] Willem J. M. Levelt. A survey of studies in sentence perception: 1970–1976. In W.J.M. Levelt and G.B. Flores d'Arcais, editors, *Studies in the Perception of Language*. John Willey & Sons, 1978.

[Miikkulainen and Dyer, 1988] Risto Miikkulainen and Michael G. Dyer. Forming global representations with extended backpropogation. In *Proceedings of the IEEE Second Annual International Conference on Neural Networks*, pages 285–292. IEEE, July 1988.

[Miikkulainen and Dyer, 1989] Risto Miikkulainen and Michael G. Dyer. A modular neural network architecture for sequential paraphrasing of script-based stories. In *Proceedings of the International Joint Conference on Neural Networks*, pages II–49–56. IEEE, 1989.

[Miikkulainen and Dyer, 1990] Risto Miikkulainen and Michael G. Dyer. Natural language processing with modular neural networks and distributed lexicon. Technical Report CSD-900001, UCLA Computer Science Dept., January 1990. Submitted to *Cognitive Science*.

[Newell, 1980] Allen Newell. Physical symbol systems. *Cognitive Science*, 4:135–183, 1980.

[Nirenburg, 1989] Sergei Nirenburg. Knowledge-based machine translation. *Machine Translation*, 4:5–24, 1989.

[Pollack, 1990] Jordan B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105, 1990.

[Pollack, 1991] Jordan B. Pollack. The induction of dynamical recognizers. *Machine Learning*, 1991. To Appear.

[Servan-Schreiber *et al.*, 1988] David Servan-Schreiber, Axel Cleeremans, and James L. McClelland. Learning sequential structure in simple recurrent networks. Technical Report CMU-CS-88-183, Computer Science Department, Carnegie Mellon University, November 1988.

[Smolensky, 1990] Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artifical Intelligence*, 46:159–216, 1990.

[St. John and McClelland, 1990] Mark F. St. John and James L. McClelland. Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence*, 46(1-2):217–257, 1990.

[Touretzky, 1990] David S. Touretzky. BoltzCONS: Dynamic symbol structures in a connectionist network. *Artificial Intelligence*, 46(1-2):5–46, November 1990.

[Van Gelder, 1990] Tim Van Gelder. Compositionality: A connectionist variation on a classical theme. *Cognitive Science*, 14:355–384, 1990.