Learning Recursive Distributed Representations for Holistic Computation

LONNIE CHRISMAN

(Received 16 July 1991; revised paper accepted 4 November 1991)

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 by Chalmers; however, the pure transformational approach does not consider the eventual inference tasks during the process of learning its representations. Confluent inference is introduced as a method for 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.*

KEYWORDS: Connectionism, holistic computation, structure-sensitive inference, distributed representations, confluent inference.

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 inability of most connectionist models to represent or make use of such structure has hindered the application of these models to higher level cognitive tasks and has been a source for attacks on the connectionist enterprise (Fodor & 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 & McClelland, 1990; Mikkulainen & Dyer, 1989; Lee *et al.*, 1990). These models usually map syntactic compositional structure into distributed representations by using various composing and decomposing functional operations. Sharkey (1991) gives a brief review of these new representational issues.

Lonnie Chrisman, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15217, USA. E-mail: chrisman@cs.cmu.edu.

The emergence of these new distributed representations for structured data creates the possibility for a new and intriguing mode of computation: *holistic inference*. This form of inference occurs in a *gestalt* fashion by deriving a solution to the inference problem directly from a representation of structured data without decomposing, locating, or accessing its constituent elements. The most interesting case revolves around the type of distributed representation that (van Gelder, 1990) characterizes as possessing *functional compositionality* without *concatenative compositionality*—i.e. 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 microstructure that emerges in the representation. This microstructure may provide a means for the efficient computation of certain classes of inference such as the *intuitive inference* described by (Hinton, 1990).

It is far from obvious that structure-sensitive holistic inference is even 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 would allow the pertinent content to be accessible in the appropriate way. On the other hand, the touted abilities of neural nets to capture relevant regularities in data may instead cause these regularities to be directly reflected in the resulting microstructure of the representations, thus opening the doors to a new realm of inference, the essence of which would be radically different from most people's conceptions of how computation must be performed.

Chalmers' 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 passification (i.e. 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 impressive 100% accuracy, thus giving an existence proof for structure-sensitive holistic computation.

The use of pure transformational inference, as employed by Chalmers, imposes a separation between the learning of representations and the training of the network that performs an inference. This prevents a system from tuning its representations for a particular set of inference tasks. When inference tasks are known in advance, it may instead be preferable to account for the inference tasks while learning representations, so that the information most relevant to the inference will be more likely to be accessible within the resulting distributed representation.

This paper introduces a method, *confluent inference*, that removes the separation between learning representations and training on the inference task. The method attempts to simultaneously embed encodings for both the input and output of an inference task within a single representation of the input. This encourages the confluence of the representations by causing commonalities with the output to be easily accessible from a representation of the input problem. For example, in experiments reported in this paper, a confluent inference training process learns representations for English and Spanish sentences. When the sentence 'He does not want it' is encoded, a distributed representation is obtained that encodes both this sentence and its Spanish translation ('No lo quiere'). Rather than only representing

the syntactic input in an alternative form, the confluent representation encodes something closer to the content of the input problem, in this case an interlingua-like representation.

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↔Spanish translation domain.

This 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. This is followed with a discussion of the characteristics that allow confluent inference to be effective, and a comparison with interlingua representation. 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 represented 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. There is no hard limit upon the maximum depth of any branch, nor is there any specific upper bound on the number of distinct trees that can be stored. In practice, however, 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; however, 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-hand L units of the input, and an *empty* or **nil** vector on the right-hand 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-hand 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, such as by including a special *stop* element at the beginning of the list.

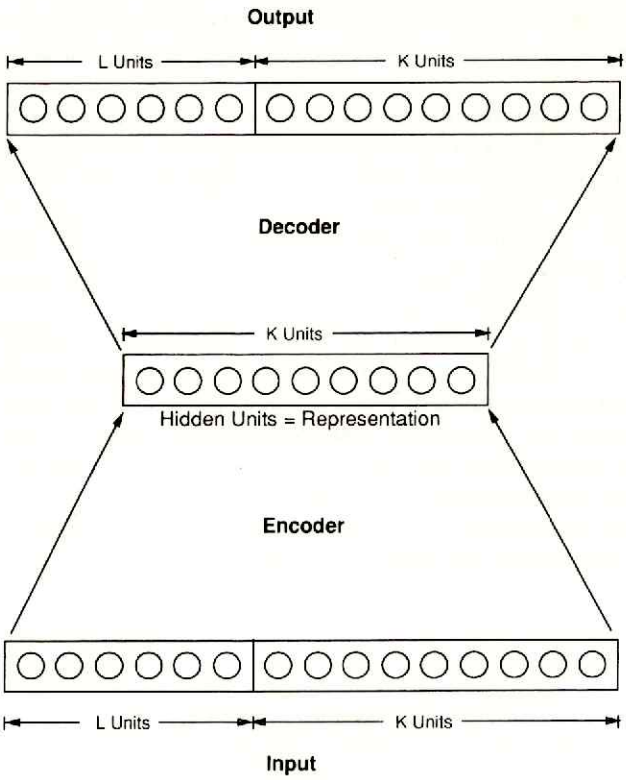

Figure 1. The sequential RAAM structure.

The network is trained to *auto-associate* the desired inputs by using the back-propagation procedure. A list element, along with the encoding of the proceeding portion of the list, is placed on the input units and the network is trained to reproduce the 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 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. None the less, 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).

Beyond being a mere structure-storage device, a RAAM's representations suggest that much more may be possible (Sharkey, 1991). In order to compress arbitrary data structures down to a fixed-width hidden layer, the RAAM must make use of regularities, and these regularities may become reflected directly in the distributed representations. The representations encode the compositional structure of the original data structure as well as additional statistically based information about these regularities. Highly efficient computation that takes advantage of this otherwise unavailable information might use the distributed representations directly. In the terminology of van Gelder (1990), the distributed representations learned by a RAAM are functionally compositional without concatenation, so that individual elements of a data structure, and relationships between elements, are not usually directly reflected in the resulting representation. Inferences that use a representation directly without accessing its compositional structure are 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 $D(r) \in X$. When $D(E(x)) = x$, we say the network is capable of auto-associating (i.e. 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 $f(\cdot)$ the translation function that converts any sentence from English to Spanish.

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

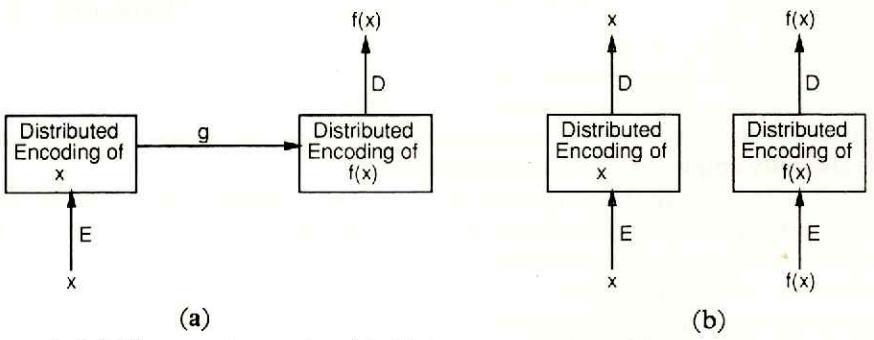

Figure 2. (a) The transformational holistic computation of $f(x)$ by $g(\cdot)$. (b) Usually the auto-association of x and $f(x)$ is used to learn representations before learning the transformation $g(\cdot)$.

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, with connection weights providing the $E(\cdot)$ and $D(\cdot)$ mappings. Chalmers 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 generates 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 (i.e. $E(\cdot)$ 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. Note that with transformation inference, the representations develop independently from the inference task(s), preventing the system from tuning its representations for a given set of inference tasks. From a cognitive-science perspective, this is unsatisfying since representations are seldom learned independently of any task. This makes pure auto-association unrealistic as a model of representation-learning processes. None the less, when tasks come along later after representations are learned, transformational inference seems particularly important.

3.2. Confluent Inference

Confluent inference provides one technique to account for the eventual inference tasks *while learning recursive-distributed representations*. The resulting representations are biased to deal with the tasks for which they were learned (cf. Miikkulainen & Dyer, 1988, 1989; St John & McClelland, 1990). This technique attempts to strike a compromise between the ease of the desired inferences and the necessary auto-associative capabilities.

Confluent inference 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 & Dyer, 1988) can be viewed in this way. In order to obtain a better understanding of confluence, the discussion and experiments focus upon the use of pure confluent inference. In a later section, the possibilities for composite configurations are considered, where confluence is used to shape the representations, and transformational inference is synergistically employed for the inference task.

If x is an input to an inference task, and $f(x)$ is the desired output, then *confluent inference* attempts to simultaneously encode both x and $f(x)$ within the representation for x. Empirically this has been observed to result in a confluence (i.e. 'coming together') of the representation for x with the representation for $f(x)$. By closely associating x and $f(x)$ in this manner, the representation for x becomes tuned for the inference task.

Taken to an extreme, confluent inference suggests that 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 (i.e. decodings), one corresponding to the initial problem, the other to the answer of the inference task. The encoding is viewed as representing a bundle of information rather than solely an encoding of the syntactic input, and inference is performed by selecting a special decoder to extract the information needed to form the answer. For the English↔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.

For clarity, the inferences under consideration in this paper will be limited to one-to-one functions; however, the confluence technique may similarly be applied to general N-to-1 functions as described in Chrisman (1991). When confluent inference is used to learn a one-to-one function, $f(\cdot)$, 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 utilized. 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 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 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). This network is capable of representing both x and $f(x)$. When $D_2(E_1(x)) = f(x)$ as in Figure 3(b), $f(x)$ 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 none the less play a critical role during the process of learning representations. The key point is that both x and $f(x)$ are combinatorial data structures with embedded constituent structure. Even if auto-association is not required by a system, the auto-associative pathways are necessary for performing the required recursive processing to build representations of whole structures from the constituent parts. 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 $y = f(x)$.

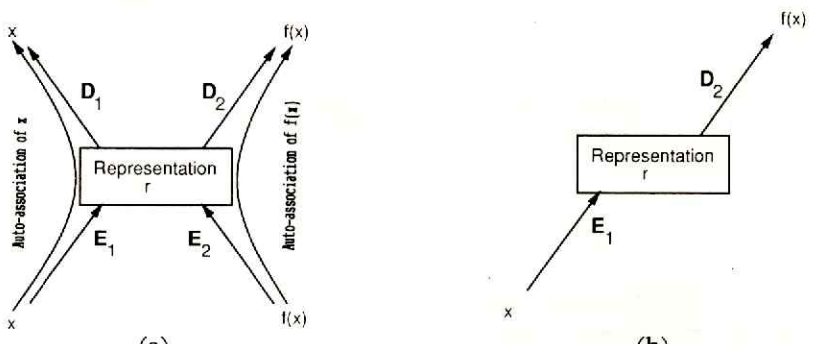

Figure 3. (a) Two-way auto-association. (b) The computation of $f(x)$ by confluent inference.

It should be emphasized that $\langle E_1(\cdot), D_1(\cdot)\rangle$ and $\langle E_2(\cdot), D_2(\cdot)\rangle$ are always distinct networks, even when the domain X and range Y of $f(\cdot)$ use the same alphabet, as was the case in Chalmers' sentence passivation task. This is a very different approach than with transformation inference, for it views a representation as a bundle of information about the content of a given problem, as opposed to viewing it simply as a syntactic encoding of the input. For sentence passivation, once a sentence is encoded, $D_1(\cdot)$ can be used to extract the active sentence interpretation, while $D_2(\cdot)$ extracts the passive one.

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 designed to test the abilities of pure confluent inference without a composite 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. 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

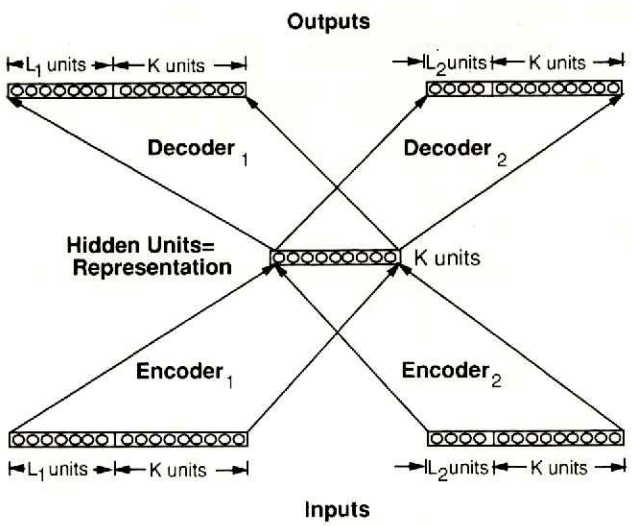

Figure 4. The dual-ported RAAM architecture.

in Figure 4 when $\langle a, {r_1}^{i-1} \rangle$ is placed on $Encoder_1$'s input. Let $Decode_1(r_j) = \langle b, r_k \rangle$ 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 $x = (x_1, x_2, \ldots, x_n)$ as input and produces a list $f(x) = (f_1, f_2, \ldots, f_m)$ as output. One epoch of the training process is as follows.

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

1. let $r_1^0 = r_2^0 = empty$.
2. for $i = 1 \ldots (n-1)$ 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 $i = 1, \ldots, (m-1)$ 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 $x, f(x)$.

The algorithm assumes a one-to-one 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 is that auto-association is required for all sublists (as with the basic RAAM), but confluent association is only required for the complete list.

The training process for the English to Spanish translation task is shown pictorially in Figure 5. For a step in the RAAM encoding process that encodes an uncompleted sentence, the normal RAAM configuration shown in Figure 5(a) is used. The words marked by the $\langle$ and $\rangle$ symbols signify that the corresponding distributed representation for that subsentence appears or is inserted in the designated location. Confluence is introduced by modifying the final step of the encoding process as shown in Figure 5(b). The final step requires the complete encoding in order to decode into both English and Spanish through $Decoder_1$ and $Decoder_2$ respectively. In the figure, the encoding for '$\langle$No tenemos$\rangle$' 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 affects the representations chosen for the encodings of earlier sentence fragments.

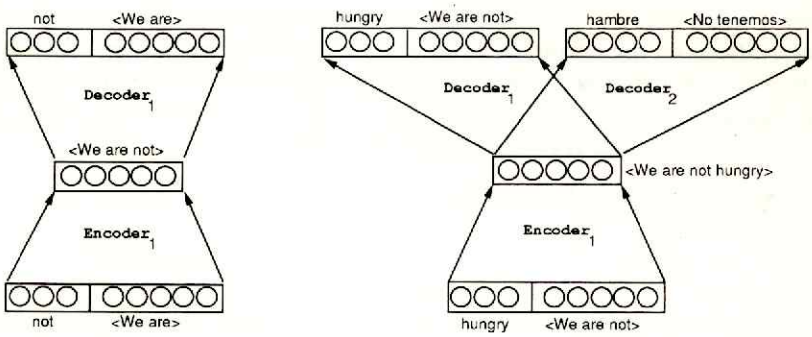

Figure 5. Training of the dual-ported RAAM. (a) The basic sequential RAAM configuration is used to auto-associate all incomplete subsentences. (b) For the completed sentence, the system is required to produce both the auto-associated output as well as the translation (or in general, the answer to the desired inference). Back-propagation is used at each step with the appropriate network configuration.

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 & McClelland (1990); Miikkulainen & Dyer (1989, 1990); and Lee *et al.*, (1990). In the predictive counterpart, a configuration similar to Figure 5(b) would consistently be employed for each subsentence. Upon seeing the first word of the sentence ('We'), back-propagation would use the complete answer ('hambre'—'⟨No tenemos⟩' as the target output for 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. When the system uses the same representations for multiple inference tasks (cf. Miikkulainen & Dyer, 1989), Figure 4 can be easily generalized to a multi-ported RAAM architecture with all inference tasks influencing the resulting representations (Chrisman, 1991).

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↔Spanish translation task. A back-propagation network was also previously applied by Allen (1987) to a small English to Spanish translation task using a multilayer feed-forward network. Unlike the structure-sensitive encodings learned by RAAMs, Allen restricted the maximum sentence length so that all words could simultaneously be 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 is non-trivial since the regularities and vocabulary in each language are distinct. When both English and Spanish sentences are auto-associated by a pure RAAM, it is possible that this distinctness may cause the RAAM to

develop unrelated encoding schemes for each language, making a holistic transformation very difficult. In fact, Allen (1987) reports encountering this phenomenon. 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'.[1] 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 (i.e. 432 total sentences) were enumerated from a vocabulary of 36 English and 36 Spanish words. These words and their (localist) encodings are given in the Appendix. 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. 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.↔Lo tiene. ('It he_has')
- We do not want it.↔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.↔Es estudiante.
- He is happy.↔Está contento.
- He is hungry.↔Tiene hambre.

Similarly, the Spanish verb 'tener' can map to different English verbs:

- Tienen razón.↔They are right.
- Tienen dinero.↔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.↔Usted es joven.
- Lonnie is young.↔Lonnie es joven.

Also, different Spanish conjugations exist for the English conjugation 'are':

- You are here.↔Usted está aquí.
- We are here.↔Estamos aquí.
- They are here.↔Están aquí.

These surface phenomena make the task particularly ill-suited for word-for-word translation. Since the set of sentences seem to require a semantic association rather than a purely syntactic association, the emergent representational microstructure 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⊕59)–40–(62⊕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 were successfully memorized, with nine words being only 'weakly' learned wherein 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 the case. The resulting distributed representations for a small sampling of the sentences is shown in Figure 6. For the entire corpus of sentences, it is clear that the representational confluence is pronounced. In fact, in 99% of the sentences (427/432), the Euclidian distance between the representation for a sentence and the representation for its translated sentence is smaller than the distance between the sentence and any other sentence in the corpus.

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 (e.g. '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 & Dyer, 1988, 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 English 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 appear to be truly distributed within the representation.

The second experiment tested generalization. The sentence pairs were randomly partitioned into two equally sized groups, and one group (of 108 sentence pairs) was used for training. A *stop* marker (all bits assigned an activation of 1.0) was prepended

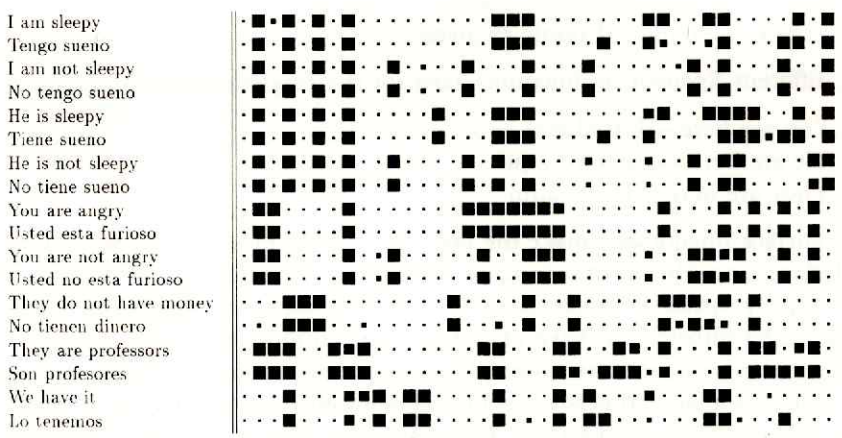

Figure 6. Distributed representations obtained during experiment 1.

to each of the training sentences, and simply treated as an additional word during the training process. A learning rate of $r = 0.1$ was used throughout. The momentum was set at $m = 0.3$ for the first 100 epoches, and $m = 0.9$ for all remaining epochs. After 3300 epochs, all 108 sentences in the training set and their translations were successfully memorized, with four words being 'weakly' learned. (These bits had activations within $\tau = 0.3$ of the correct value.) The remaining 108 sentence pairs (216 sentences) were then used for testing the generalization accuracy of the system. Each sentence was decoded until the stop marker was produced, and was compared against the correct reproduction. The generalization accuracy is divided into two parts, the auto-associative accuracy (i.e. the ability to encode and decode a sentence to obtain the original sentence) and the translational accuracy (i.e. the ability to correctly translate a sentence into the other language). The trained network correctly auto-associated 89% (192/216) and correctly translated 75% (161/216 = 85/108 English→Spanish and 76/108 Spanish→English) of the testing sentences. For these figures, only the sentences that were reproduced exactly were counted as correct. Of the incorrect sentences, 87% (21/24 and 48/55 respectively) were almost correct, differing either by a single erroneous word or an incorrect subject with a consistent verb conjugation.

While 25% of the testing sentences were translated incorrectly in the previous experiment, roughly half this many (10%) 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 (i.e. represent) the sentences. 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, shown earlier in Section 4, 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 3500 training epochs (using the same learning parameters as the previous experiment), the network had perfectly memorized all 432 auto-associations as well as the 216 translations from the training set. The remaining 108 sentence pairs were then used to test the generalization accuracy of translation. The system translated 89% (191/216 = 96/108 English→Spanish and 95/108 Spanish→English) of the sentences perfectly. Of the mistakes, 60% (15/25) 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. Table I shows all testing sentences that were incorrectly translated by the network in this third experiment.

A fourth experiment was performed in order to obtain a basis for comparison with transformational inference. Two RAAM networks, one for English and one for Spanish, were trained to auto-associate the entire corpus of 216 sentences in each language using network topologies of 62–40–62 and 59–40–59 respectively. Both

Table I. Incorrectly generalized translations in the third experiment

Testing sentence	Erroneous translation		Correct answer
Lo tengo	*I* want it	*	I have it
No lo tengo	*I* do *not* want *it*	+	I do not have it
I am not a professor	*No son estudiante*	+	No soy profesor
I am not a student	*No son estudiante*	+	No soy estudiante
I am not from California	No soy de *Pittsburgh*	*	No soy de California
Tiene dinero	He *is sleepy*	*	He has money
Es de Pittsburgh	*He are from* Pittsburgh		He is from Pittsburgh
He is not from Pittsburgh	No es de *California*	*	No es de Pittsburgh
No es de Pittsburgh	*Professors happy is* not from Pittsburgh		He is not from Pittsburgh
Lonnie is from Pittsburgh	Lonnie es de *California*	+	Lonnie es de Pittsburgh
Usted no lo quiere	*You does* not want it		You do not want it
Usted tiene dinero	You *have they*		You have money
You are a professor	Usted *tiene quiere* profesor		Usted es profesor
Usted es profesor	*Lonnie* do *a professor*		You are a professor
You are a student	Usted tiene *quiere estudiante*		Usted es estudiante
Usted es estudiante	*Lonnie do a student*		You are a student
You are not a student	Usted no *tiene estudiante*		Usted no es estudiante
Usted no es de California	*Lonnie student* not *from* California		You are not from California
You are not from Pittsburgh	*Lonnie* no es de *Pittsburgh*	*	Usted no es de Pittsburgh
Usted no es de Pittsburgh	*Professor Lonnie happy* not *from* Pittsburgh		You are not from Pittsburgh
They do not have it	No lo tenemos	*	No lo tienen
Tienen dinero	They *are young*	+	They have money
We want it	Lo quieren	*	Lo queremos
Tenemos dinero	*We are money*		We have money
We do not have money	No *tiene* dinero	*	No tenemos dinero

Words in italics were only weakly activated with at least one output unit between 0.2 and 0.8. Asterisks indicate responses in the main training set and plus signs those which appeared in the auto-association set.

networks were fully trained in 500 epochs using $m=0.9$ and $r=0.1$. Next, the same 108 sentence pairs used in Experiment 3 for confluent inference training were used to train a transformation network to convert from the English representation of a sentence into the Spanish representation. Using the Spanish RAAM, this representation was then recursively decoded until a stop marker was obtained, and the decoded sentence was compared with the actual translation. Training began with $m=0.3$ and $r=0.1$, with the momentum increased to $m=0.9$ after 100 epochs. Achieving 100% accuracy on the training set was difficult, with little noticeable improvement after 3000 epochs, even when the learning rate was decreased to $r=0.02$ and the momentum increased to $m=0.95$. Training was stopped after 20 000 epochs, at which point the network was able to reproduce all of the training sentences, with 16 of the reproduced words containing 'weak' bits with activations between 0.2 and 0.8. Generalization was then tested on the remaining 108 sentence pairs. The network successfully translated 71% (77/108) of the English sentences perfectly (as compared with 89% by confluent inference in experiment 3). Of the incorrectly translated sentences, 94% (29/31) could be considered almost correct, differing only by a single word or by an incorrect subject with a matching verb conjugation. Since the network topologies, training set, and testing sets were

identical in the third and fourth experiments, this seems to be the most meaningful experimental comparison possible. Confluent inference achieved slightly better than half the translation error rate; however, this single experimental comparison should not be taken as a representative ratio of performance for natural language translation tasks overall since the sentence corpus is relatively small and possibly not representative of more realistic translation tasks.

6. 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 has empirically been observed to appear in the form of very similar representations for a problem and its answer. To determine the appropriate scope for the confluent technique, it is necessary to understand when the confluence of the two representations is likely to occur.

The confluence 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. It is these commonalities in high-level or 'deep' meaning that confluent inference attempts to bring into the representations. 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, confluent inference may be useful for leveraging that overlap, while transformational inference may be more appropriate for the remainder of the task. Composite approaches are considered in a later 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)).

6.1. Portability

In many systems, it is likely that new inference tasks may come along after the system has learned its representations. In these cases it may be appropriate to train for the new inference tasks without altering representations. With pure confluent inference, this requires training an additional decoder to map directly from a fixed representation of the input to the desired (structured) output. With a transformational component it consists of training a network to convert from a given fixed

representation of the input problem to the given fixed representation for the output. *Portability* refers to the ability of a representation to support inference tasks that are introduced after representations have been formed (Lee *et al.*, 1990).

Like confluent inference, FGREP has also been used to form representations that are appropriate for given inference tasks (Miikkulainen & Dyer, 1989). However, FGREP and confluent inference are very different with respect to portability. 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 & Dyer 1990), the representations for each of these words becomes identical. For this reason, Lee *et al.* (1990) argue for the development of task-independent representations, claiming that because distributed semantic representations (DSRs) are learned independently of any particular task, they are *portable* to tasks outside of the training environment.

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 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.

6.2. Interlingua

Confluent representations in the language translation experiments 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, i.e. 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. However, perhaps the most blatant difference is that the confluent representations are not absolutely constrained to be identical for the same sentence in two different languages—confluence is an emergent property that has been observed to occur empirically. The precise representation for a sentence may vary by small amounts depending upon the source language (Figure 6). These variations 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 that 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, pp. 21–47).

For tasks other than natural language translation, the analogy to interlingua becomes more vague. As we consider variations (e.g. N-to-1 mappings (Chrisman, 1991)) on the inference task, it is hard to identify any relevant relationship between confluence and interlingua.

7. Composite 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, 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. Additionally, when an inference substantially alters the semantic content within a problem, or when inferences are chained in a sequence of steps, then a transformational component appears necessary. In this section, two possible approaches are presented for obtaining a synergistic combination of transformational and confluent inference.

In a composite 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 composite 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 composite 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 (given earlier in Section 4). Therefore, a natural point to turn off confluent inference would be 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 7(a). However, after confluent inference is turned off, the two representation spaces are treated as distinct as shown schematically in Figure 7(b). Two different encoder–decoder pairs are still used.

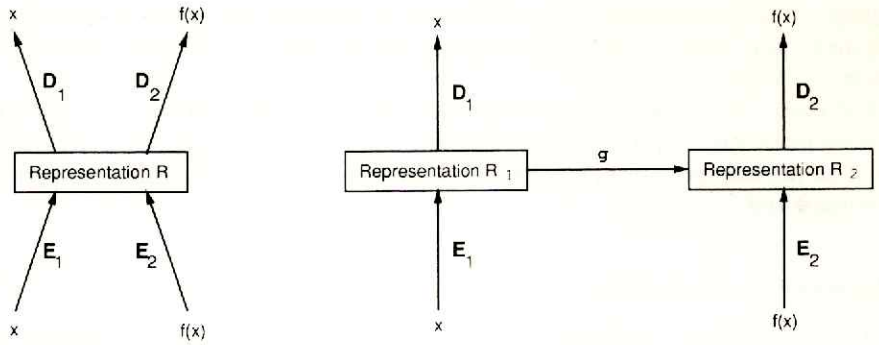

Figure 7. Decoupled composite architecture. (a) While confluent inference is on, the representation spaces for a problem and its answers are shared. (b) After confluent inference is turned off, the two representation spaces are treated as if they are separate and distinct.

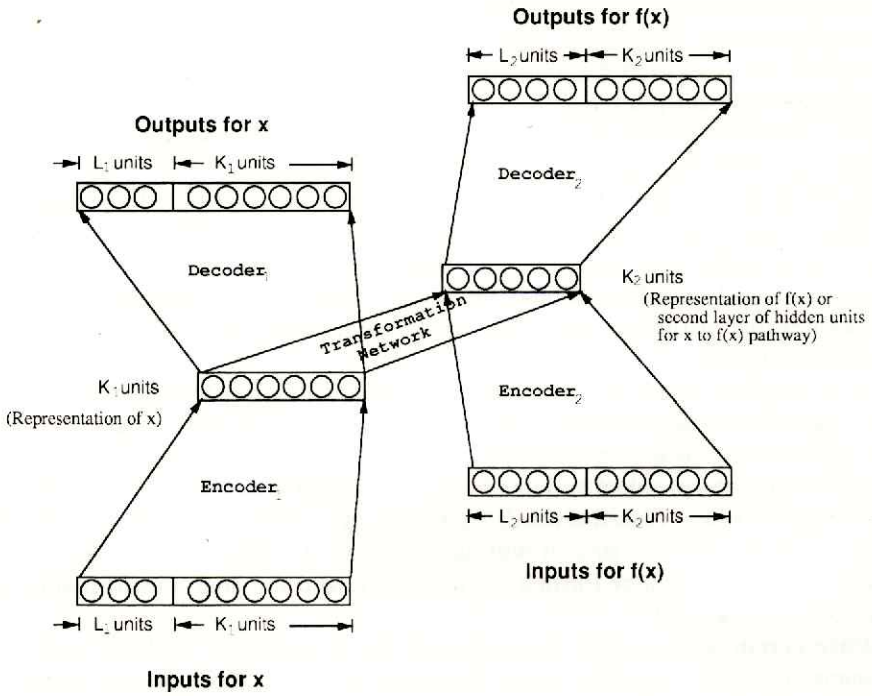

Figure 8. Coupled composite architecture.

This 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 composite 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 composite architecture is shown in

Figure 8. 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 of this configuration is similar to that of the dual-ported RAAM except that the confluence of $f(x)$ (step 5 in the confluence training algorithm) is not enforced, and the second set of representation units are treated as an extra hidden-layer by back-propagation during the confluence step (step 4).

It should be evident that in both composite approaches, the inverse of a one-to-one 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.

8. Conclusions

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 microstructure 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 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 does not tailor the representations for the inference task. A second form of holistic inference, *confluent inference*, was introduced in order to achieve this capability. 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↔Spanish translation task, by using pure confluent inference the system perfectly translated 89% of the testing sentences that were not in its training set. 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 David Chalmers, Noel Sharkey and Dave Touretzky for helpful and influential comments on early drafts of this paper. This research was sponsored by

NASA under Contract NAGW-1175. The views and conclusions contained in this article are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the US government.

Notes

1. However, Nora Celis (personal communication, 1991) has obtained encouraging positive results using RAAM-based transformational inference for English to Spanish translation.
2. As Figure 4 shows, the topology is similar to that of two different networks with topologies of 62–40–62 and 59–40–59 which share the hidden same hidden units. The ⊕ notation summarizes this.

References

Allen, R. B. (1987) Several studies on natural language and back-propagation. In M. Caudill & C. Butler (Eds), *Proceedings of the IEEE First International Conference on Neural Networks*, pp. II–335–341. New York: IEEE.

Blank, D. S., Meeden, L. A. & Marshall, J. B. (1992) Exploring the symbolic/subsymbolic continuum: a case study of RAAM. In J. Dinsmore (Ed.), *The Symbolic and Connectionist Paradigms: Closing the Gap*. Hillsdale, NJ: Lawrence Erlbaum.

Chalmers, D. J. (1990) Syntactic transformations on distributed representations. *Connection Science*, **2**, 53–62.

Chrisman, L. (1991) Learning recursive distributed representations for holistic computation. *Technical Report CMU-CS-91-154* (Carnegie Mellon University, Pittsburgh, PA). An earlier expanded version of this paper.

Drozdek, A. (1989) Interlingua in machine translation. *Proceedings of the 17th Annual ACM Computer Science Conference*, p. 434. New York: ACM Press.

Elman, J. L. (1990a) Finding structure in time. *Cognitive Science*, **14**, 179–212.

Elman, J. L. (1990b) Structured representations and connectionist models. In G. Altmann (Ed.), *Computational and Psycholinguistic Approaches to Speech Processing*. New York: Academic Press.

Fodor, J. A. & Pylyshyn, Z. (1988) Connectionism and cognitive architecture: a critical analysis. *Cognition*, **28**, 3–71.

van Gelder, T. (1990) Compositionality: a connectionist variation on a classical theme. *Cognitive Science*, **14**, 355–384.

Goodman, N. (1983) *Fact, Fiction, and Forecast*, 4th edn, Chap. III. Boston, MA: Harvard University Press.

Hinton, G. (1986) Learning distributed representations of concepts. *Proceedings of the 8th Annual Cognitive Science Society Conference*, pp. 48–54. Hillsdale, NJ: Lawrence Erlbaum.

Hinton, G., McClelland, J. & Rumelhart, D. (1986) Distributed representations. In D. Rumelhart, J. McClelland & the PDP Research Group (Eds), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1, Foundations*, Chap. 3, pp. 77–109. Cambridge, MA: MIT Press.

Hinton, G. E. (1990) Mapping part–whole hierarchies into connectionist networks. *Artificial Intelligence*, **46**, 47–75.

Lee, G., Flowers, M. & Dyer, M. G. (1990). Learning distributed representations of conceptual knowledge and their application to script-based story processing. *Connection Science*, **2**, 313–345.

Levelt, W. J. M. (1978) A survey of studies in sentence perception: 9170–1976. In W. Levelt & G. F. d'Arcais (Eds), *Studies in the Perception of Language*. New York: Wiley.

Miikkulainen, R. & Dyer, M. G. (1988) Forming global representations with extended backpropogation. *Proceedings of the IEEE Second Annual International Conference on Neural Networks*, pp. 285–292. New York: IEEE.

Miikkulainen, R. & Dyer, M. G. (1989) A modular neural network architecture for sequential paraphrasing of script-based stories. *Proceedings of the International Joint Conference on Neural Networks*, pp. II–49–56. New York: IEEE.

Miikkulainen, R. & Dyer, M. G. (1990) Natural language processing with modular neural networks and distributed lexicon. *Technical Report CSD-900001* (UCLA Computer Science Dept.). Submitted to *Cognitive Science*.

Nirenburg, S. (1989) Knowledge-based machine translation. *Machine Translation*, **4**, 5–24.

Pollack, J. B. (1990) Recursive distributed representations. *Artificial Intelligence*, **46**, 77–105.

Pollack, J. B. (1991) The induction of dynamical recognizers. *Machine Learning*, **7**, 227–252.

Servan-Schreiber, D., Cleeremans, A., & McClelland, J. L. (1988) Learning sequential structure in simple recurrent networks. *Technical Report CMU-CS-88-183* (Computer Science Department, Carnegie Mellon University).

Sharkey, N. E. (1991) Connectionist representation techniques. *AI Review,* **5,** 143–167.

Smolensky, P. (1990) Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence,* **46,** 159–216.

St John, M. F. & McClelland, J. L. (1990) Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence,* **46,** 217–257.

Touretzky, D. S. (1990) BoltzCONS: dynamic symbol structures in a connectionist network. *Artificial Intelligence,* **46,** 5–46.

Appendix A: Word Encodings

Tables AI and AII show the word encodings used during the natural language translation experiments described in Section 5. The first group of bits encode word

Table AI. The English word encodings

	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
does	1	0	0	0	0	0	0	0	0	0	1	0	1	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
wants	1	0	0	0	0	0	0	0	0	0	1	0	0	1	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
has	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	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
is	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	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
I	0	1	0	0	0	0	0	0	0	1	0	0	1	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
he	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
Lonnie	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	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
we	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	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
from	0	0	0	1	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
a	0	0	0	0	0	1	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
angry	0	0	0	0	0	0	1	0	0	0	0	0	0	1	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
here	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	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
old	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	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
sleepy	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
hungry	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
thirsty	0	0	0	0	0	0	1	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
professors	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	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
students	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0
money	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	1	0
Pittsburgh	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
California	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

Table AII. The Spanish 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
Lonnie	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

category (i.e. verb, subject-noun, pronoun, preposition, adverb, determiner, adjective, simple-noun, or location). The middle group of bits encode a rough intuitive notion of plurality. For verbs, these represent 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.