CONNECTIONIST REPRESENTATIONS FOR NATURAL LANGUAGE: OLD AND NEW

Noel E. Sharkey
Department of Computer Science
University of Exeter

Connectionist natural language processing research has been in the literature for less than a decade and yet it is already claimed that it has established a whole new way of looking at representation. This article presents a survey of the main representational techniques employed in connectionist research on natural language processing and assesses claims as to their novelty value i.e. whether or not they add anything new to Classical representation schemes.

Connectionist natural language processing (CNLP) research has barely been in existence for a decade (cf. Sharkey & Reilly, in press, for a potted history) and yet it has grown enough to attract criticism from some formidable guardians of the Classical tradition. For example, Fodor and Pylyshyn (1988) claimed that connectionist representations could work for NLP if and only if they were only Classical representations exhibit the properties of compositionality, and structure sensitivity and therefore only Classical representations can be used for natural language processing. While it is not the purpose of this paper to address the Fodor and Pylyshyn arguments in detail, some of their arguments will be used to examine connectionist representations for their novelty value. The main aim of the paper is to present a critical survey, and the Classical criticisms are discussed in this light of the survey. The stance taken here will be that there are novel connectionist representational types which are compositional (though not in the Classical sense) and which can be manipulated by structure sensitive operations.

Natural language research is normally concerned with two main types of representation: structural or syntactic representation and semantic or meaning representation. The latter is usually divided into the representation of lexical items and the representation of larger units such as phrases or sentences. In much connectionist work it is difficult to separate syntactic and semantic representation. Nonetheless, each of the different types will be discussed in turn and a taxonomy will be proposed.

1. The representation of meaning and structure.

1.1 Semantic representations.

Localist v Distributed.

One of the major debates in connectionist research of the early to mid-eighties was concerned with whether or not individual items in a net should be represented by the activity on a single unit in a net - a *localist* representation (e.g. Cottrell, 1985) - or whether their representation should be a *distributed* pattern of activation across a number of units (e.g. Hinton, McClelland, and Rumelhart, 1986). Localist connectionism became almost synonymous with Jerry Feldman's group at Rochester, USA, while the proponents of distributed representations resided in San Diego (UCSD) as Rumelhart and McClelland's Parallel Distributed Processing group (c.f. Feldman, 1989 for a fuller discussion).

Hinton (1989) points out that terms *localist* and *distributed* are relative terms.

implementation. We can extract two simple defining criteria for distributed representations from the Hinton paper. First, an entity that is described by a single term in the descriptive language is represented by more than one element in a connectionist implementation. For example, if the letter 'F' is a term in the descriptive language, then the distributed elements in the descriptive language may be the features 'f', 'f', and 'f'. Second, each of the elements in the connectionist implementation must be involved in representing more than one entity described by a single term in the descriptive language. For example, the features that make up the letter 'F' may also be used as part of the representation for the letter 'E'.

Figure 1 shows a fairly typical example of a localist net from the Rochester school (Shastri & Feldman, 1986). This is rather like the old semantic network idea in which each unit in the net represents a single concept and is linked to other units by positive or negative weights. In most of the early Rochester work the weights were set by hand rather than by a learning algorithm. But there is no reason why local representations cannot be trained using the same algorithms as those of the distributed school.

Both representational types have their advantages and disadvantages. The advantage of localist representation is its transparency. Each unit is clearly labeled and so it is easy to see what its function is in the network. However, it is difficult to see what the novelty value of such representations amounts to. Since each unit represents a single semantically interpretable symbol, there is no new action that does not appear in the Classical tradition. Connectionists using such purely local representations must rely on the novelty value of the processing implementation, which is the main thrust of their research¹.

As we shall see later, despite their seeming opacity, there are advantages to distributed representation which make them more desirable. Unlike local

¹We have not discussed here the problems of building a representational theory using purely local representations for whole propositions. Such a theory would have to make the unlikely assumption that mind has a finite number of propositions which can never be unpacked and used to construct novel propositions (see Fodor & Pylyshyn, 1988).

representation, there are number of types of distributed representation. Two broad classes will be discussed here: symbolic and subsymbolic (c.f. Smolensky, 1988). All other types may be subdivided into these two groups.

Symbolic v Subsymbolic.

To understand the distinction between symbolic and subsymbolic representations, we need to look first at the notion of a *microfeature*. This is a term that has not been used entirely consistently in the literature. All would agree that microfeatures are the atomic elements in a distributed connectionist representation. However, some authors (e.g. McClelland & Kawamoto, 1986) use the term to refer to individual elements which are semantically interpretable on their own without examining their role in the representation e.g. propositional predicates such as *is human*, *is soft*. These sort of microfeatures are symbolic in the sense that they refer to properties in the world. That are much akin to semantic features, and are sometimes called semantic localist.

Figure 2 shows some of the microfeatures used by McClelland and Kawamoto (1986). While these are closely related to earlier semantic feature representations, they have the defining criteria for a distributed representation. That is, a single term in the descriptive language, such as the word 'ball', is represented by a number of microfeatures in the connectionist implementation i.e. non-human, soft, neuter, small, compact, rounded, unbreakable, food. In addition, the microfeature representing the word 'ball' are shared by other words. For example, 'cheese' share non-human, soft, neuter, small, and rounded.

Feature Dimensions & Values

NOUNS	
HUMAN	human, nonhuman
SOFTNESS	soft, hard
GENDER	male, female, neuter
VOLUME	small, medium, large
FORM	compact, 1D, 2D, 3D
POINTINESS	pointed, rounded
BREAKABILITY	fragile, unbreakable
OBJ-TYPE	food, toy, tool, utensil, furniture animate, nat-inan
VERBS	
DOER	yes, no
CAUSE	yes, no-cause, no-change
TOUCH	agent, inst, both, none, AisP
NAT-CHGE	pieces, shreds, chemical, none unused
AGT-MVMT	trans, part, none, NA
PT-MVMT	trans, part, none, NA
INTENSITY	low, high

Other authors (e.g. Hinton, 1981; Smolensky, 1988) use the term microfeature to refer to individual elements that are semantically uninterpretable (without participating in further processing) or subsymbolic. By this we mean that no one individual microfeature refers to a property in the world. Rather, reference to such properties emerges from a pattern of activation across several microfeatures. This style of representation is more like how many imagine information to be encoded in the nervous system. Each neuron is an unlabelled unit in a large collective from which symbolic information emerges.

One disadvantage of using symbolic microfeatures is that the task of choosing a sufficient set of microfeatures is in the hands of the researcher. This can be problematic in that it is difficult to determine, *a priori*, what microfeatures would be required for a given task. At its worst, the use of symbolic microfeatures can lead to the sort of *ad hoc* "tuning" from which much of AI research has suffered i.e. run the system and, if it doesn't work, try some different microfeatures (though it may be possible to circumvent part of this problem by conducting an empirical investigation with humans to determine a sufficient set of microfeatures).

With semantically uninterpretable microfeatures, these problems need not occur. It is possible for a net to develop a sufficient set of semantically uninterpretable microfeatures for a required task² (e.g. Milkkulainen & Dyer, 1987).

Some advantages of Distributed representations

Distributed representations require less memory than localist ones. More distributed items can be represented per vector element (for vectors with more than two elements). A classic example is McClelland and Rumelhart's (1981) representation of the 26 letters of the alphabet with a 16 element vector of visual features. A localist scheme would require a 26 element vector.

Localist networks can encode up to n items, where n is the dimension of the representation space; while distributed networks have the capacity to encode $2^n - (n+1)$ items. In Example 1, a comparison is given, of localist representations versus distributed representations using a four-bit vector. Note that the localist vector holds only 4 items while the distributed vector holds 11.

Localist representations
1000 0100 0010 0001

Distributed representations
1100 1010 0110 1110 1001 0101 1101 0011 1011 0111 1111

Example 1. Comparisons of a distributed versus localist representation on a four-bit vector.

The difference in storage capacity becomes more apparent as the size of the representing vector gets larger as shown in Table 2. With only 10 bits, 101 distributed representations may be encoded, whereas a localist representation will have a storage capacity of only 10 items.

No Bits	Localist	Distributed
2	2	1
3	3	4
4	4	11
5	5	26
6	6	57
7	7	120
8	8	247
9	9	502
10	10	1013

Table 2. Comparisons of the storage capacity for localist and distributed systems.

Another important advantage of distributed representations is that they have built generalisation properties. In localist representations, all of the vectors representing items are, by definition, perpendicular to one another and equidistant (Hamming Euclidean distance). Thus it is difficult to capture similarities and differences between items in localist representation space (although it can be done by explicit marking). On the other hand, distributed representations can form a denser representation space. For example, for simplicity of exposition, imagine that a set of distributed representation vectors are unit normalised (i.e. are all set to length 1). These vectors may then be described geometrically as points on a unit hypersphere as illustrated on the sphere in Figure 4.

Figure 4. Two clusters of vectors, a and b, are shown on the surface of a unit sphere.

The point here is that similar items will cluster on the surface of the hypersphere. The clusters are shown at a and b in Figure 4. It is then relatively easy to develop a process model in which similar items produce similar or identical results. For example, if a net was trained to take microfeatural representations of HORSE, CAT, and COW as input and map them onto ANIMAL in the output, then we would expect a microfeatural representation for DOG, which was not in the training set, to produce the response output ANIMAL. That is, we would expect the vector representation for DOG to be sufficiently close to the vector representation for other animals to have a similar effect.

A third advantage of distributed representations for CNLP is that they provide a natural basis for content addressable memory (e.g. Hopfield, 1982). That is, a network can be trained such that given a partial description (a subset of microfeatures), it will complete the pattern. Sharkey (1989c) has taken advantage of this property to "fill in" information not explicit in a text. This is the connective equivalent of default reasoning, but it comes automatically as a standard feature of distributed representations.

In summary, the various types of semantic representations for CNLP have been examined here. It was also proposed that the most powerful representation, in terms of memory efficiency, pattern completion and storage efficiency, was the distributed subsymbolic. But there is another important reason for favouring subsymbolic representations. Their examination represents a research topic that is unique to connectionism. Distributed symbolic representations have been applied in the Classical tradition in areas such as speech recognition. However, as shall be argued later, the study of subsymbolic representation is a new departure.

1.2 The Representation of Structure

A distinction can be drawn between those connectionist structures which are syntactically accessible and those which are syntactically implicit. Syntactically explicit representations are those in which structural operations rely on the actual spatial layout of the elements in the representations. Syntactically implicit representations, on the other hand, are not spatially concatenative and do not contain explicit representations of their constituent tokens. This distinction will become clearer as the different styles of structural representation are discussed in turn.

Syntactically structured representations

A common form of structural representation in AI is the sentence *frame* (e.g. Minsky 1975)³. In this notation, propositions or concepts are described as structures explicitly containing a number of slots that have constraints on what items may fill them. For example, Schank (1972) developed the notion of conceptual dependency in which there were a small number of action frames (approximately 12). For example,

John drove mary to the station.

would be represented as:

JOHN $\Leftarrow \Rightarrow$ PTRANS $\dashrightarrow$ MARY $\left\{ \begin{array}{l} \rightarrow \text{STATION} \\ \leftarrow \text{?HOME?} \end{array} \right.$

where ?HOME? is a default value. This can also be represented as a frame with slots

agent	action	object	to	from
JOHN	DROVE	MARY	STATION	?HOME?

Hinton (1981) described a distributed representation for propositions which shares number of properties with these sentence frames. In Hinton's system, binary vectors representing distributed propositional triples are conceptually divided into three parts. The elements of the n^{th} partition, by analogy with frames, represent all and only the permissible fillers of the n^{th} slot. Thus the only constraint on what item may fill a slot is only that the appropriate vector partition has bits for representing the items. There are defaults for filling in missing values in the partitions/slots, but these fall out of the pattern completion process in Hinton's system.

These vector frames are syntactically explicit because the vector partitions slots in a structured frame⁴. Thus it is easy to tell at a glance what are the constituents. Probably for this reason, vector frames have been used widely (McClelland & Kawamoto, 1986; St. John & McClelland, in press; Touretzky & 1988). Their main use is as input and output buffers to make the inputs and comprehensible.

Although very useful, vector frames suffer from three particularly bad problems. First, there can be considerable redundancy in the representations. For example, most items that could appear in an Object partition could also appear in the Position partition, and so they have to be represented twice by different elements. The second problem relates to the first in that the representation for the same item in the two partitions is entirely different. Thus the system has no way of "knowing" that the *book* in the Object partition is the same as the *book* in the Position partition. A third problem with vector frames is that they have a fixed length or a fixed number of partitions. Thus all of the input sentences can be only of that length.

A number of ways have been found to get around this fixed length restriction. One is having a processing window that moves along the input vector (e.g. Sejnowski & Rosenberg, 1986). Other researchers have taken the alternative approach of employing recurrent networks (e.g. Elman, in press) which accept sequential input. We shall return to examine these representations in more detail in the section on *Encoding temporal structure*.

The vector frame representation, it could be argued (c.f. Fodor & Pylyshyn, 1987), is simply a connectionist implementation of symbolic case frames. By being implemented in a connectionist way, vector frames add nothing new to the theory of language cognition. For a connectionist representation to add something new it must be fundamentally different from classical representations. Nonetheless, vector frames are useful as input and output representations. They can act as a *symbol surface* or as a window into connectionist representations can emerge for the researcher to check out what has been happening underneath. We now turn to examine distributed representations of syntactic structure which are syntactically implicit.

Syntactically unstructured representations.

Saying that a representation is syntactically implicit means that it does not contain explicit constituent structure. The most common form of syntactically implicit representations are those that result from a mapping of an input space onto a space of lower dimensionality. For example, Hinton (1981) mapped propositional networks onto a lower dimensionality PROP assembly using fixed random weights. The network, in a sense, recruits a set of PROP units to represent it in a syntactically implicit form. Through a learning process, it is possible to map the PROP activity back onto the higher dimensional Triple space, and thus recreate the structure. Coarse coding, as Hinton called it, is discussed at length in Hinton, McClelland & Rumelhart, (1986).

Variations of this type of compact representation are common in the literature (Touretzky & Hinton, 1988; Touretzky and Geva, 1987; Willshaw & von der Malsburg, 1979; Cottrell, Munro, and Zipser, 1989) and may be set up by a simple algorithm in conjunctive coding (e.g. McClelland & Kawamoto, 1986), or may be learned

⁴A similar technique was employed in McClelland and Rumelhart's (1981) model of word recognition. The vector partitions in that instance were used to represent positional information of the input.

by supervised (e.g. Hinton, 1986) or unsupervised techniques (e.g. Kohonen, 1982). Regardless of the learning technique used, the representation encodes statistical regularities of the input (usually) by reducing the pattern environment to a low dimensional feature space. When required, the lower dimensional coding can be decoded onto the symbol surface again.

To make the notion of compact representations clearer, from the perspective of both semantic and structural representation, we turn now to briefly analyse one of the learning algorithms in more detail.

1.3 Representation in a back propagation net

In this section, we discuss how the generalised delta learning rule constructs representations. This is perhaps the most commonly employed learning algorithm in connectionist natural language research. We begin by discussing its application in feedforward net architecture with two layers of weights (as shown in Figure 5).

Figure 5. An illustration of a standard back propagation net with 2 layers of weights. The circles represent the units and the lines between are partial representation of the weights.

Before running the learning, all of the weights, from the input units to the hidden units and from the hidden units to the output units, are usually set to small random values in the range -1 to $+1$ ⁵. In the forward operation of the net, the input vector is set to the binary states of the first input pattern. This vector is then mapped onto the hidden unit vector $\mathbf{h}$ (normally of lower dimension than $\mathbf{v}$) by multiplying $\mathbf{v}$ by the first weight matrix $\mathbf{W}_1$ and applying the squash function $S: \mathbf{W}_1 \mathbf{v} \rightarrow \mathbf{h}$ (where $S = 1/(1+e^{-x})$, $x = \mathbf{W}_1 \mathbf{v}$). Then $\mathbf{h}$ is mapped onto the output vector $\mathbf{o}$ using the same squash function $S: \mathbf{W}_2 \mathbf{h} \rightarrow \mathbf{o}$.

During learning $\mathbf{o}$ is compared with a target vector $\mathbf{t}$ to determine its correctness. If $\mathbf{t} - \mathbf{o} > 0$ then an error correction procedure is set in motion which adjusts the weights matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ such that $\mathbf{o}$ is closer to $\mathbf{t}$. The mathematical rationale of the weight adjustment have been given full treatment in many sources (e.g. Rumelhart, Hinton, and Williams, 1986; Hinton, 1989) and so will not be repeated here.

⁵A smaller range of initial values is sometimes used. Kolen and Pollack (1990) demonstrate the importance of initial conditions to learning in monte carlo simulations and show that under certain

What we are interested in at present is how the representations develop over time. Our first question must then be: where are the representations. Up until now, our concern has only been with representations that are patterns of activation across a set of units. In this sense, the hidden unit activations are the representations discussed in sections 1.1 and 1.2), while the lower weights are part of the encoding function ($S:W_1v \rightarrow h$) and the upper weights, are part of the decoding function ($S:W_2h \rightarrow o$).

However, we may also describe the encoding and decoding weights as representing themselves. It is instructive to view the learning process geometrically to get an intuitive grasp of the notion of weight representation. The first step in learning is to adjust the upper (decoding) weights so that the weight vectors for output units that want to be 'on' are moved closer to the current vector of hidden unit activation, while the weight vectors for outputs that want to be 'off' are moved away from the current hidden unit vector. Secondly, the lower (encoding) weights are adjusted to put the vector of hidden unit activations even closer to the weights whose outputs should be 'on' and further away from weights whose outputs should be 'off'.

The upshot of this learning is: (i) input patterns that are required to produce a particular set of outputs will learn to produce similar hidden unit activations and thus they will learn to have similar 'projective' weights; (ii) similar output patterns will have to have similar 'receptive' weights. It is possible to examine this similarity using a Euclidean distance metric, where the distance between two vectors v_1 and v_2 in R^n is the length of $v_1 - v_2$ i.e. distance $d = ||v_1 - v_2||$, where length $||v|| = \sqrt{v \cdot v}$. Euclidean distances can then be fed into a cluster analysis program which plots the similarities on a 2D dendrogram.

The point to be made here is that it is not just unit activations that may be represented under the representational umbrella. The weights can also be thought of as representing the input and output representations. It can be argued that the projective weights from the inputs to the hidden units are representations of individual elements and the hidden unit activations are a compositional representation of strings of the individual elements.

2. Recent issues in natural language representation.

2.1 Encoding Temporal Structure

One problem for researchers employing the standard feedforward back propagation networks discussed in 1.3, has been how to represent temporal sequences. In reading and speech understanding, the input is structured in time, and thus the behavior of a system cannot be determined solely on the basis of the current input element. What is required is some sort of memory for previous elements in a sequence (or some way to be combined with the current element). Up until now, the input representations that have been examined involve presenting each whole sequence to the system as a single input. This is equivalent to buffering the input stream until a sequence has been completed, before acting on it. The question then reverts to how to structure the contents of the buffer.

The main approach examined here has been the vector frame (e.g. Hinton, McClelland & Rumelhart, 1981). Another approach is that of Rumelhart & McClelland (1986). They adapted Wicklegren's (1969) proposal for the representation of words as sequences of context sensitive phoneme units (Wicklephones). They represent each phone as the phone itself, its predecessor, and its successor (e.g. the vowel in the word "cat" would be represented as **kaɪ**). Thus a set of overlapping

Wicklefeature is a single unit that conjunctively coarse codes a feature of the central phoneme, a feature of its predecessor, and a feature of its successor. A different method, employed by Sejnowski and Rosenberg (1986) for their NETtalk system, employed a window containing 7 letters that moved across an input text. The central element of the window, on each successive move was encoded using the three elements on either side of it as context.

An alternative solution to the encoding of sequential structures, without using a buffer, was proposed by Elman (1988) with the introduction of a network architecture for predicting successive elements of a sequence (sentence). This is a variant of the feedforward multi-layer perceptron which allows feedback or recurrent links from the hidden units to the input. As each element of a sequential structure, such as a sentence, is coded onto the input units, the previous hidden unit vector is copied onto memory units in the input stream⁶. In this way, the meaning of an element of a sequence will be shaded by the context of the prior elements. In a sense each input cycle contains a memory of the previous cycles in the sequence.

Elman (1989) has conducted a number of simulations using the simple recurrent net architecture (SRN). He presented short sentences to the net, one word at a time, using the next word as a target. Thus the task for the net was to predict the next word in a sentence. Elman found that the network had developed hidden unit representations for the input patterns that reflected information about the possible sequential ordering of the inputs e.g. the net knew that the lexical category VERB followed the lexical category NOUN. Cluster analyses of the hidden unit activations revealed that the verb category is broken down into those verbs which require a direct object and those for which a direct object is optional. Furthermore, the analyses showed that the nouns were divided into animates and inanimates with a further subdivision for human and non-human. In a larger scale analysis, Elman also discovered that the tokens of particular types clustered together⁷. Thus, hidden unit representation in the simple recurrent net, after learning, can be shown to exhibit a number of properties needed for a lexical category structure and type/token hierarchies.

Elman (1989) also investigated the representation of grammatical structure in a study which used a phrase structure grammar to generate the input sentences. This grammar allowed recursion through the use of a relative clause category that expanded to NPs that permitted further relative clauses. The results suggest that the net had learned to represent abstract grammatical structure. For example, when presented with a subject noun the net correctly predicted a verb which agreed with the number of the subject noun (i.e. singular/plural), even when a relative clause intervened. In addition, given a particular noun and verb, the net was shown to correctly predict the class of the next transition allowed by the grammar, thus demonstrating the representation of verb argument structure. Finally, the results from the recursive representations showed limitations. These representations were found to degrade after about three levels of embedding.

The same type of SRN was employed by Servan-Schrieber, Cleeremans, and McClelland (1989) in a study which involved learning a finite-state grammar. There were many interesting results from this study. But the most important results, for

⁶Elman's recurrent net is actually a variant of Jordan's (1986) sequencing net. Jordan took his recurrent links from the output units or from the training vector to the input units whereas Elman's recurrent links are from the hidden units to the input units.

⁷The representation of abstract grammatical structure has proved to be a difficult task for

our purposes, are: (a) the net learned to be a perfect recogniser for a finite grammar (at least for the Reber (1967) grammar they used). (b) under conditions, long distance sequential dependencies were exhibited, even embedded sequences. The latter result was best when the dependencies were re at each step. Moreover, performance across embedded strings deteriorated length of the string increased.

In sum, by extending the backpropagation algorithm in a simple recurrent net, been possible to add a number of features to the compact representations that discussed in Section 1. Primarily, SRNs allow the representation to encode sequential information such as the order of the input, and the path from one element to another. They also exhibit a certain ability to allow the encoding of long range sequential dependencies across embedded sequences.

2.2 Recursive distributed representations

One aspect of natural language processing that has been problematic in the connectionist community is that natural languages are recursive. We have seen, in the last section, how this posed difficulties for SRNs. In this section, we discuss two recent attempts at representing recursive structures.

Tensor product representations

The tensor product system (Smolensky, in press) combines lexical items with syntactic roles in a way which is mathematically equivalent to *outer product* (cf. Sharkey, 1989). That is, a vector representing an item (or role filler), $\mathbf{i}$, is combined with a vector representing a role, $\mathbf{r}$ by the outer product $\mathbf{ir}^T$. This is a tensor of rank 2 and results in a square matrix of activations. However the formalism goes beyond the simple outer product in that it enables the construction of recursive representations for, say, syntactic trees by using 3rd, 4th, or n th order tensors. A third order tensor is a cube of unit activation and orders beyond the third are hypercubes.

There are two main problems with tensor products. First, with deep embedded representations could grow exceedingly large. Second, when the input vectors (role fillers for the roles) are not orthogonal the tensorial representations have to be constructed by more complex incremental learning methods (e.g. the delta rule, linearly independent pattern sets, or back propagation). This makes the process less manageable as it is not at all obvious how such learning would take place⁸. However, processing and memory considerations aside, Smolensky presents an elegant and formally tractable theory of recursive representation. We can leave to later research to work out how to develop it in real time and how to use it for recognition.

Recursive auto-associative memory (RAAM)

Hinton (1988) outlined an idea for handling embedded clauses by inserting a description of them into larger representations. However, he did not detail a method by which such representations could be learned. This challenge has been taken up by Pollack (in press) who shows how such a reduced description can be learned using Recursive Auto-Associative Memory (RAAM). The RAAM architecture is the standard feedforward net with two layers of weights (for encoding and for decoding the hidden unit representations) and the standard back propagation algorithm.

employed for learning. Pollack has shown the power of the RAAM system for encoding a sequential stack with PUSH and POP and also for encoding and decoding syntactic trees. The whole trees are represented in a single layer of hidden units and can be decoded in cycles until the terminal symbols appear as the outputs. The difference between RAAM and the usual back propagation net rests on the method for presenting the input patterns.

We shall briefly describe the operation of a RAAM system here using the example of a simple binary tree: ((A B) (C D)). First the input space is divided into n partitions, with k units in each partition. The size of n depends directly on the maximum valency of the tree to be represented (in our simple example $n = 2$). Since this is an autoassociative net the output vector is identical to the input vector; both have n units and there are k hidden units.

The representation of the binary tree would be formed as follows: (i) A and B are presented in the two vector partitions and autoassociated. The resulting hidden unit representation R_1 is kept to one side (on an external stack or something); (ii) C and D are presented and autoassociated and the resulting hidden unit representation R_2 is put to one side; (iii) R_1 and R_2 are presented as input and autoassociated. The resulting hidden unit vector R_3 is a representation of the entire tree. R_3 can be decoded by presenting it directly to the hidden units and the outputs will be R_1 and R_2 . These are then presented in turn until the terminals have been decoded.

Pollack (in press) presents a range of interesting simulation results which show RAAM to be a very effective method for encoding and decoding recursive structures. The only problem is that the method of presentation of inputs relies on an external stack and it is not altogether clear what a pure connectionist implementation of this would be. However, regardless of how the representation is constructed, Pollack has demonstrated how unstructured representations can encode recursive representations in a compact form.

2.3 Compositionality and structure sensitivity

In Section 1, connectionist representations were classified into different types. Some of these, as we have seen, are very similar to their Classical counterparts in that they contain explicit symbol tokens and/or have concatenative constituent structure (e.g. localist concept nodes, symbolic microfeatures, vector frames), and some are weak (e.g. localist proposition nodes). It is not the aim here to cast doubt on the value of the research using these representation schemes, but to consider whether or not the representations themselves (not the research) have novelty value.

From the review above, it should be quite clear that compact subsymbolic connectionist representations are different than Classical syntactic structure representation. This style of representation, Fodor and Pylyshyn (1988) argue, is not compositionally structured. However, as Van Gelder (1990) points out, Fodor and Pylyshyn are implicitly discussing only one type of compositionality: *spatial concatenative composition*. In this mode of composition, the spatial layout of the symbols (reading from left to right) is important (indeed crucial) for symbol manipulation and inference. Van Gelder states that for a mode of combination to be concatenative, "... it must preserve tokens of an expression's constituents (and the sequential relations among tokens) in the expression itself."

In contrast, to Classical concatenative representation, the type of compact connectionist representation we have been discussing may be considered to have a different mode of combination. That is, "pure" connectionist representations are not concatenative, but are functionally compositional nonetheless. It is worth quoting Van Gelder again on this point. "We have functional compositionality when there

general, effective and reliable processes for (a) producing an expression given constituents, and (b) decomposing the expression back into those constituents. Connectionist models can certainly perform (a) and (b) as well as meet the criterion that the processes must be general, effective, and reliable. By *general*, van Gelder means that that the process can be applied, in principle, to the construction and decomposition of arbitrarily complex representations. We have seen how a simple feedforward back propagation net can learn to encode and decode representations. To be *effective* the processes must be mechanistically implementable and to be *reliable* they must always generate the same answer for the same inputs. Once a connectionist net has finished learning it meets both of these criteria.

Given that connectionist representations are functionally compositional, the question is: do such seemingly unstructured representations carry structural information? And a subsidiary, though perhaps more important, question is: do these representations allow direct structure sensitive operations? The short answer to the first question is obviously "yes". Even in the early Hinton (1981) model of semantic nets, the vector frames of structured input representations were coarse coded or compact representation such that they could be accurately reconstructed onto an identical vector frame. To be reconstructed, the coarse coded representation must have been carrying structural information. In fact, they were carrying information about concatenative structure without themselves being concatenative.

The subsidiary question, as to whether connectionist representations allow structure sensitive operations, is partly addressed by the answer to the previous question. However, it might be argued that even the functionally compositional connectionist representation may be a variation on the Classical theme because the connectionist representations must emerge onto the symbol surface before they can be structurally manipulated. For example, Fodor & McLaughlin (1990) claim that in order to support structure sensitive operations, compositional representation must contain explicit tokens of the original constituent parts. This position has been subjected to a rigorous empirical investigation by Chalmers (in press) which refutes it.

Chalmers constructed compact recursive distributed representations of syntactic structures using Pollack's (in press) RAAM system (described in 2.2 above). After training the net to develop compact representations for both active and passive sentence structures, Chalmers set out to test the structure sensitivity of the compact representation. He did this by attempting to train the transformation of the compact representation of active sentences into the compact representation of the passive sentences. The experiment was successful in that it demonstrated that connectionist representations can be structurally manipulated (passivisation) without recourse to emergence onto the symbol surface.

3. Conclusions

The main classes of connectionist representation for natural language processing have been examined in this paper. For convenience these were divided into semantic representations (Section 1.1) and structural representations (Section 1.2). In Section 1.1, semantic representations were classified into major types: localist and distributed and a number of advantages were pointed out for distributed representations (memory efficiency, content addressability, and built-in generalisation). In addition, two flavours of distributed representation were pinpointed: symbolic and subsymbolic. On the question of the novelty of connectionist semantic representations

the subsymbolic was shown to be the only contender. Distributed symbol representation have a lot of similarities with Classical feature theory.

On the syntactic side, a distinction was drawn between representations which are syntactically explicit and syntactically implicit. It was argued that only the latter could be considered to be representationally novel. The syntactically implicit representations were discussed further in Section 2.3. It was argued that they were functionally compositional (as opposed to concatenative) and could be sensitive to structural manipulations without recourse to decoding into the original symbol tokens of their constituent parts.

This paper displays optimism about the development and utility of unique connectionist representations i.e. subsymbolic, syntactically implicit representation. We have seen only one connectionist study in which these representations have been shown to be structure sensitive. However, this is just the beginning. We have also seen (Section 2.1) how non-concatenative distributed representations can carry information about temporal structure, long distance dependencies, lexical category structure and the type/token distinction. We have also seen how they can represent finite state grammars. In section 2.2, we saw how research on connectionist representation had begun to overcome one of the hardest problems for CNLP, the representation of recursive structures.

All in all, despite (and to some extent thanks to) Fodor and Pylyshyn's critique of connectionist representation, it looks as though the prognosis for CNLP is good. Judging by the explosion of research we have seen up until now, the next few years are expected to yield many exciting new results.

References.

Chalmers, D.J. (in press) Syntactic Transformations on Distributed Representation. *Connection Science*, 2.1.

Cottrell, G.W. (1985) A Connectionist Approach to Word Sense Disambiguation. Ph.D. Thesis, TR154, Department of Computer Science, University of Rochester, NY.

Cottrell, G.W., Munro, P. & Zipser, D. (1989) Image Compression by Backpropagation: An Example of Extensional Programming. In N.E. Sharkey (Ed) *Models of Cognition: A Review of Cognitive Science*. Norwood, N.J.: Ablex.

Dolan, C.P. & Smolensky, P. (1989) Tensor Production System: a Modular Architecture and Representation. *Connection Science* 1(1), 53-68.

Elman, J.L. (1988) Finding the structure in time. TR 880, CRL, University of California, San Diego.

Elman, J.L. (1989) Representation and structure in connectionist models. TR 890, CRL, University of California, San Diego.

Feldman, J.A. (1989) Neural Representation of Conceptual Knowledge. In N.E. Sharkey (Ed) *Models of Cognition: A Review of Cognitive Science*. Norwood, N.J.: Ablex.

Fillmore, C.J. (1968) The Case for Case. In E. Bach and R. Harris (Eds) *Universals of Linguistic Theory*. Holt, Rhinehart & Winston.

- Fodor, J.A. & McLaughlin, B. (1990) Connectionism and the Problems of Systematicity: Why Smolensky's Solution Doesn't Work. *Cognition*, 35, 183-204.
- Fodor, J.A. & Pylyshyn, Z.W. (1988). Connectionism and Cognitive Architecture: A Critical Analysis. *Cognition*, 28, 2-71.
- Hinton, G.E. (1981) Implementing Semantic Networks in Parallel Hardware. In G.E. Hinton & J.A. Anderson (Eds) *Parallel Models of Associative Memory*. Hillsdale, N.J.: Lawrence Erlbaum.
- Hinton, G.E. (1986) Learning Distributed Representations of Concepts. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*.
- Hinton, G.E. (1989) Connectionist Learning Procedures. *Artificial Intelligence*, 41, 184-235.
- Hinton, G.E., McClelland, J.L. & Rumelhart, D.E. (1986) Distributed Representations of Words and Derivations. In D.E. Rumelhart & J.L. McClelland (Eds) *Parallel Distributed Processing, Volume 1*. Cambridge, MA: MIT.
- Hopfield, J.J. (1982) Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences, USA*, 79, 2554-2558.
- Hornik, K., Stinchcombe, M. & White, H. (1989) Multilayer Feedforward Network as Universal Approximators. *Discussion paper 88-45R*. Department of Economics, UCSD.
- Jordan, M.I. (1986) Attractor Dynamics and parallelism in a Connectionist Sequence Machine. *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, Amherst, MA, 531-545.
- Kohonen, T. (1982) Clustering, Taxonomy, and Topological Maps of Patterns. In J. Lang (Ed) *Proceedings of the Sixth International Conference on Pattern Recognition*. Silver Spring, MD: IEEE Computer Society Press.
- Kolen, J.F. & Pollack, J.B. (1990) Back Propagation is Sensitive to Initial Conditions. *Technical Report 90-JK-BPSIC*. Ohio State University.
- McClelland, J.L. & Kawamoto, A.H. (1986) Mechanisms of Sentence Processing: Assigning Roles to Constituents. In J.L. McClelland & D.E. Rumelhart (Eds) *Parallel Distributed Processing Volume 2*. Cambridge, MA: MIT.
- McClelland, J.L. & Rumelhart, D.E. (1981) An Interactive Activation Model of Letter Perception: Part I. An Account of Basic Findings. *Psychological Review*, 88, 375-407.
- Miikkulainen, R. & Dyer, M. G. (1987) Building Distributed Representations with Microfeatures. *Technical Report UCLA-AI-87-17*, AI Laboratory, Computer Science Department, University of California at Los Angeles, CA.
- Minsky, M. (1975) A Framework for Representing Knowledge. In P.H. Winston (Ed) *The Psychology of Computer Vision*. New York: McGraw-Hill.
- Pollack, J.B. (in press) Recursive Distributed Representations. *Artificial Intelligence*.
- Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). Learning Internal Representations by Error Propagation. In D.E. Rumelhart & J.L. McClelland

- bbins, A. (1989) The Distributed Representation of Type and Category. *Connection Science* 1.4.
- hank, R.C. (1972) Conceptual dependency: a theory of natural language understanding. *Cognitive Psychology*, 3, 552-631.
- gnowski, T.J. & Rosenberg, C.R. (1986) A Parallel Network that Learns to Read Out loud. *Technical Report JHU/EECS-86/01*. John Hopkins University.
- ervan-Schreiber, D., Cleermans, A., & McClelland, J.L.(1988) Encoding sequential structure in simple recurrent nets. TR CMU-CS-88-183. Computer Science Department, Carnegie-Mellon University.
- harkey, N.E. (1989a) The Lexical Distance Model and Word Priming. *Proceedings of the Eleventh Cognitive Science Society*.
- harkey, N.E. (1989b) Lexical Representations for Prepositional Attachment. *Report of British Telecom CONNEX Project*.
- harkey, N.E. (1989c) A PDP Learning Approach to Natural Language Understanding. In I. Aleksander (Ed) *Neural Computing Architectures*. London:North Oxford Academic.
- harkey, N.E. & Reilly, R. (in press) An overview of connectionist natural language processing. *Parallel update*.
- hastri L. & Feldman J.A. (1986) Neural Nets, Routines and Semantic Networks. In N.E. Sharkey (Ed) *Advances in Cognitive Science*. Chichester: Ellis Horwood.
- molensky, P. (1989b) Distributed Representations of Symbolic Structures constructed with Tensor Products. *Artificial Intelligence* (in press).
- molensky, P. (1988) On the Proper Treatment of Connectionism. *Behavioral and Brain Sciences*, 11, 1-74.
- molensky, P. (in press) Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems. *Artificial Intelligence*.
- st John, M.F. & McClelland, J.L. (in press). In R. Reilly & N.E. Sharkey (Eds) *Connectionist Approaches to Language Processing*. Hillsdale, N.J.:LEA.
- ouretzky, D.S. & Geva, S. (1987) A Distributed Connectionist Representation for Concept Structures. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, 155-164.
- ouretzky, D.S. & Hinton, G.E. (1988) A Distributed Connectionist Production System. *Cognitive Science* 12(3), 423-466.
- van Gelder, T. (1990) Compositionality: a Connectionist Variation on a Classical Theme. *Cognitive Science*, 14.
- Wickelgren, W.A. (1969) Context-sensitive coding, associative memory, and serial order in (speech) behavior. *Psychological Review*, 76, 1-15.
- Willshaw, D.J. & von der Malsburg, C. (1979) A Marker Induction Mechanism for the Establishment of Ordered Neural Mapping: Its Application to the Retino-tectal Connections. In *Philos. Trans. Roy. Soc. Lond. B* 287, 203-243.