# Connectionism and Compositional Semantics

**David S. Touretzky**

May 1989

CMU-CS-89-147

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

## Abstract

Quite a few interesting experiments have been done applying neural networks to natural language tasks. Without detracting from the value of these early investigations, this paper argues that current neural network architectures are too weak to solve anything but toy language problems. Their downfall is the need for "dynamic inference," in which several pieces of information not previously seen together are dynamically combined to derive the meaning of a novel input. The first half of the paper defines a hierarchy of classes of connectionist models, from categorizers and associative memories to pattern transformers and dynamic inferencers. Some well-known connectionist models that deal with natural language are shown to be either categorizers or pattern transformers. The second half examines in detail a particular natural language problem: prepositional phrase attachment. Attaching a PP to an NP changes its meaning, thereby influencing other attachments. So PP attachment requires compositional semantics, and compositionality in non-toy domains requires dynamic inference. Mere pattern transformers cannot learn the PP attachment task without an exponential training set. Connectionist-style computation still has many valuable ideas to offer, so this is not an indictment of connectionism's potential. It is an argument for a more sophisticated and more symbolic connectionist approach to language.

This research was supported by National Science Foundation grant EET-8716324, and by the Office of Naval Research under contract number N00014-86-K-0678.

# Connectionism and Compositional Semantics

**David S. Touretzky** [1]

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

## 1 Introduction

Natural language is an ideal source of problems for connectionism because the domain is so rich, yet for simple sentences processing takes place nearly instantaneously. Although natural language understanding obviously requires inference, interesting results have been obtained with connectionist models which do not perform what I call true dynamic inference. In the first half of this paper I define a hierarchy of connectionist models, ranging from categorizers and associative memories through pattern transformers and dynamic inferencers. The second half of the paper examines a particular natural language problem in detail: prepositional phrase attachment. I will show that PP attachment is critically dependent on compositional semantics, and that this in turn depends upon dynamic inference. The search for a connectionist solution to the PP attachment problem must therefore begin with a satisfying account of dynamic inference.

## 2 A Hierarchy of Models

Connectionist models can be organized into a hierarchy of classes based on the complexity of the input/output pairs they use as training data. In increasing power, the classes are: categorizers, associative memories, pattern transformers, and dynamic inferencers. Even the simplest class of models has interesting properties, such as parallel operation and use of distributed representations, which we may wish to incorporate into natural language models.

### 2.1 Categorizers

Categorizers place their input into one of a fixed, disjoint set of categories. They share the following properties:

- The number of output patterns is fixed, and small relative to the number of possible inputs. There is a unique output pattern for each of the *n* categories that must be recognized.
- There are many examples of each category, so each output pattern is encountered many times in the training set.

[1]An earlier version of this paper appeared as "Connectionism and PP Attachment" in D. Touretzky, G. Hinton, and T. Sejnowski (eds), *Proceedings of the 1988 Connectionist Models Summer School*, published by Morgan Kaufmann.

- The training set densely samples the space of possible inputs. No portion of the space can be safely neglected when a categorization task is nontrivial.

The simplest output representation for a neural network categorizer is a set of $n$-bit vectors with exactly one bit on in each. In competitive learning models with a winner-take-all layer, this output format is intrinsic to the architecture, but in other models it is merely an option. More complex output patterns can easily be derived from a competitive model by adding another layer of output units. Each of the winner-take-all units can then produce an arbitrary fixed output pattern by means of appropriate weighted connections to the new output layer. An example is Hecht-Nielsen's counter-propagation architecture (Hecht-Nielsen, 1987).

Several natural language problems have been posed as categorization tasks. The NETtalk grapheme-to-phoneme system (Sejnowski & Rosenberg, 1987), billed as "a model that learns to read aloud," pronounces text by sequentially processing it in seven-letter chunks. The pronunciation of the middle letter in each chunk is determined by categorizing the entire pattern into one of 55 phoneme classes. Each class has an associated output pattern, 26 bits long, that codes for phonetic features such as voicing and place of articulation. Although NETtalk appears to be addressing two highly complex language tasks, reading and speech production, it is really just learning to solve 26 binary categorization problems in parallel. (Note that the 26 problems are not solved independently because the output units share the same hidden layer.)

Whether a network is a categorizer or a pattern transformer is determined by its input/output behavior, not its architecture. Back propagation can produce either categorizers or pattern transformers. A network can be taught to simulate the behavior of a simpler class, e.g., a pattern transformer can simulate a categorizer, but many architectures are incapable of implementing the more complex classes. Classic perceptrons, for instance, can only be categorizers. Also, models that employ a competitive winner-take-all layer, such as ART (Grossberg, 1987), the Nestor learning algorithm (Reilly et al., 1982), and counter-propagation with a single winner (Hecht-Nielsen, 1987) are necessarily limited to categorization.

## 2.2 Associative Memories

Associative memory models such as (Willshaw, 1981), and recurrent associative networks such as Hopfield nets (Hopfield, 1982), are really just a special class of categorizers. Consider a hetero-associator that maps input patterns $I_j$ to output patterns $O_j$. Each stored pair $\langle I_j, O_j \rangle$ forms a class. Given an input $I^*$ that is close to some $I_j$, the associator produces the corresponding $O_j$. The output pattern $O_j$ thus serves as a *name* for the class of inputs that are sufficiently close to $I_j$ that they can evoke $O_j$.

In an auto-associator the i/o pairs are of form $\langle I_j, I_j \rangle$. When the model corrects and completes a partial pattern $I^*$ by settling to the nearest $I_j$, it is classifying the pattern $I^*$ as belonging to the category $I_j$.

Associative memories differ from categorizers in the constraints they place on the categories that can be learned. A Hopfield net treats inputs that are close in Hamming space as instances of the same class; if a few bits are missing in an input pattern they can easily be filled in by auto-association. Categorizers built by back propagation aren't constrained to treat nearby points as similar. They can use their hidden layers to learn difficult discriminations, such as parity or the Penzias two clumps/three clumps problem, where the output is critically dependent on the value of every single input bit, and points close in Hamming space often do not fall in the same class.

On the other hand, backprop nets aren't as well suited as Hopfield nets to reconstructing stored patterns from a partly corrupted version. In the Hopfield net the stored patterns are energy minima; the net will settle into one of these minima (generally the closest one) from any starting point. So a Hopfield net is predisposed to reproduce exactly the patterns it was trained on. (This is not an absolute constraint; Hopfield nets do exhibit false memories.) In contrast, when a backprop net trained on some input/output pairs is given a partially corrupted input it is most likely to produce a corrupted output, unless it has been explicitly shown how to correct corrupted patterns.

## 2.3 Pattern transformers

Pattern transformation tasks are more computationally demanding than categorization. They have the following characteristics:

- The number of output patterns grows exponentially with the size of the input. It is not fixed as in categorization.
- Only a few of the output patterns the network must produce will be familiar. Most will never have been encountered in the training set.

The ability to produce correct outputs for novel inputs is known as *generalization*. In order for the transformation task to be learnable, the function $f$ that maps inputs to outputs must be inducible (with high probability) from the training data. See Valiant (1984) for a discussion of the conditions under which this is feasible.

Pattern transformation in neural nets is easiest when similar inputs should produce similar outputs. That is, given a novel input $I^*$ that is close to some training exemplar $I_j$, the network should be required to produce an $O^* = f(I^*)$ that is close to $O_j = f(I_j)$.

One of the classic uses of pattern transformation in connectionist natural language processing is the Rumelhart & McClelland (1986) verb learning model, which transforms the phonetic representation of the root form of an English verb into a phonetic representation of the past tense. The model learns to produce correct regular past tense forms of novel verbs, and can also handle special cases such as "eat/ate" or "sing/sang". The use of a distributed, coarse coded representation for the input and output phonetic sequences helped make the correct transformation function for regular cases easy to induce from the training set.

Perhaps the most interesting application of pattern transformation to natural language is the McClelland and Kawamoto (1986) case role assignment model. In this model the input pattern is a representation of the surface form of a sentence, and the output pattern is a set of fillers for four case roles. Words are represented as vectors of semantic features. The mapping of a surface element is highly dependent on meaning and context. For example, a surface subject can fill either the Agent role (as in "the boy broke the window"), the Patient role ("the window broke"), or the Instrument role ("the hammer broke the window"). The fourth role is called Modifier; an example is the noun "sauce" in "the pasta with sauce."

For novel sentences that are similar to the training sentences (e.g., train on "the girl hit the boy," test on "the boy hit the girl"), the McClelland and Kawamoto model correctly infers the case roles of the various constituents using a combination of semantic feature and word order information. It also performs lexical disambiguation, e.g., by mapping the pattern for "bat" to either "baseball bat" or "flying bat" depending on context. A finer sort of discrimination is evident in its handling of "chicken," which may be mapped to either "cooked chicken" or "live chicken" depending on the case role it fills. Finally, the model makes some interesting generalizations by inventing new output patterns. Its training set includes sentences involving balls, which have the semantic feature *soft*, and sentences about breaking things, where the instrument (such as a hammer) is always hard. When the model is presented with "the ball broke the window" it outputs a novel feature vector for "ball" with the semantic feature *hard*.

The problem with pattern transformers is that they require an unreasonable amount of training data in order to generalize correctly. When the transformation function to be induced involves very high order predicates (in the Minsky and Papert *Perceptrons* sense), so that inputs nearby in Hamming space do not necessarily result in nearby outputs, the training set must include almost every possible input/output pair. Two examples are Allen's models of anaphora resolution and sentence translation (Allen, 1987). These models were both constructed by training a back propagation network on virtually the entire input space. The language translation model, for example, was trained on 99% of a set of 3300 English/Spanish sentence pairs generated by a simple context free grammar. It then showed good (but not perfect) generalization on the remaining 33 sentences.

## 2.4 Dynamic inferencers

A scheme that requires training on a substantial portion of all possible inputs can neither explain human linguistic performance nor produce a practical natural language understanding automaton. Training set size is exponential in the length of the input, and it becomes unmanageable extremely quickly.

The way I propose to get around this problem is by abandoning the idea of mapping the surface level representation of a sentence to its meaning in a single parallel step. Instead we need to exploit the compositionality of language: the property that the meanings of complex inputs are composed from the meanings of simpler ones (Fodor & Pylyshyn, 1988). We can exploit this property by building a model that combines portions of its input to produce novel meaning structures as intermediate internal states; these then undergo further processing as they combine with more inputs. We avoid the need to train the model on a sizable fraction of all possible inputs by introducing specialized cognitive primitives that allow it to correctly combine intermediate states never previously encountered.

The nature of these primitives remains a central question of cognitive science. Two candidates are variable binding and pointer following (what Newell (1980) calls distal access.) The latter refers to the ability to use a symbol (a compact mental object) to refer to some concept which, in its full detail, is not compact. Given a symbol we must have some way of accessing its meaning, but also, given a concept that is to be composed with others, we must have some way of generating a symbol that points to it. These symbols are meaningful in their own right, so that they can function as "reduced description" (Hinton, 1988).

If this is beginning to look suspiciously like ordinary symbol processing, it may not be such a bad thing. The compositionality issue *has* to be addressed; it will not go away. But this does not mean that connectionist networks are reduced to implementing Lisp-like symbol manipulation primitives. Connectionism still has many distinctive properties that may be important for building natural language understanders. Learning algorithms that develop distributed representations and generalize, flexible matching by parallel constraint satisfaction, and efficient parallel search are some of the things that differentiate connectionist models from ordinary Lisp code.

Formally, dynamic inferencers have the following properties:

- Input patterns are sequences of tokens from some vocabulary $V$. The number of input patterns is exponential in the length of the input, i.e., $O(V^L)$.
- Training set size is at most polynomial in $V$.
- The computational process dynamically brings together two or more pieces of information which have not been seen together before in order to derive new information. The model therefore exhibits novel intermediate states in response to novel inputs.

Notice that nothing was said above about the output patterns of a dynamic inferencer. They don't matter; it's the intermediate states that are important. Consider a network with an internal model of a static blocks world scene. Its only task is to output "yes" or "no" in response to sentences such as "Is the blue block in front of the red pyramid a large block". This is just a binary categorization task: sentences are placed either in class "yes" or class "no". But to assure a subexponential training set the problem must be implemented as a dynamic inference task. This allows the the *meaning* of the query to be derived on the fly by composing the meanings of its components, producing successively more complex internal states. To summarize:

- Dynamic inferencers can produce an exponential number of internal states, corresponding to their exponential number of possible input patterns.
- The internal states produced for complex inputs are systematically related to those for simpler inputs (compositionality property).
- This systematicity is reflected in the structure of the network, through specialized primitives for things like variable binding and distal access. The actual set of primitives required remains an open question.

There aren't yet any connectionist dynamic inferencers. However Touretzky and Hinton's distributed connectionist production system comes close (Touretzky & Hinton, 1988). It has an exponential number of working memory states, it dynamically retrieves elements from work memory and uses their components to construct new states, and it requires no training at all. But its version of compositionality is weak; working memory elements cannot be combined into more complex elements.

Derthick's $\mu$KLONE system (Derthick, 1988) is another close approximation to a dynamic inferencer. It dynamically combines information from its knowledge base with information supplied in a query. This combination is a complex process since parts of the query may conflict with previously stored facts; the network must find a plausible interpretation that violates as few constraints as possible. $\mu$KLONE extensively exploits structuring of knowledge: concepts have semantic features and are organized into a hierarchy; concepts have roles whose fillers are other concepts; roles also have semantic features and are organized into a role hierarchy; and the choice of fillers of a role is governed by value restrictions. However, $\mu$KLONE answers queries in a single parallel annealing; its internal states do not get composed to produce more complex states.

## 3 The Prepositional Phrase Attachment Problem

To reiterate, the two main points of this paper are, first, that valuable experiments in natural language understanding can be done with connectionist models that are computationally weak. These models are toys, but are interesting nonetheless because they operate in parallel, are constructed automatically by learning algorithms instead of crafted by hand, generalize successfully to novel inputs, and use distributed representations. But the second point is that implementing a real natural language understander requires more than this. It requires dynamic inference.

To illustrate both points I will discuss in detail a restricted version of an important natural language problem: determining where to attach a prepositional phrase when there are several nouns available. The version of the problem considered here is syntactically trivial but semantically rich. Inputs consist of a head noun phrase $NP_0$ followed by some number of prepositional phrases $PP_i$, each of which contains a noun phrase $NP_i$. Noun phrases consist of a definite article "the" followed by a singular concrete common noun; lexical ambiguity is excluded.. A typical example is (1):

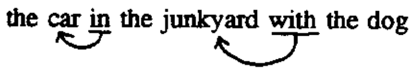
(1) the car in the junkyard with the dog

Here $NP_0$ is "the car" and there are two PPs that need to be attached to NPs.

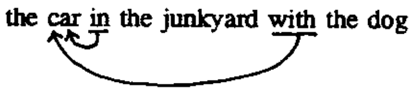
### 3.1 Syntactic constraints on attachment

An inviolable constraint of English is that PPs may only attach to NPs that precede them, i.e., the arrows showing attachment may run to the left but not to the right. Obviously then, in (1) $PP_1$ can only attach to $NP_0$. But $PP_2$ ("with the dog") can attach to either $NP_0$ or $NP_1$. A second, weaker constraint on attachment is that in the absence of other influences, PPs prefer to attach to the *nearest* NP to their left. So the preferred attachment for (1) is (2a), but (2b) is also legal, and might be preferred in some contexts.

(2a) the car in the junkyard with the dog

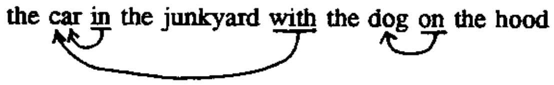


(2b) the car in the junkyard with the dog



Since the first PP of (2a) attaches to $NP_0$ and the second PP attaches to $NP_1$, we call (2a) a 0-1 attachment structure; (2b) is a 0-0 attachment structure. We can shift the attachment of "the dog" from "the junkyard" to "the car" by adding a third PP, as in (3):

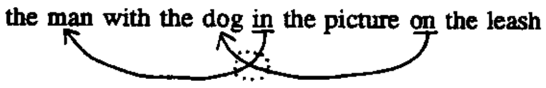
(3) the car in the junkyard with the dog on the hood



The only reasonable place to attach "on the hood" is "the dog." Since "on the hood" makes an implicit part-of reference to the car, attaching it to the dog ties the dog more strongly to the car than to the junkyard, overriding the nearest-left-neighbor attachment preference.

The most important syntactic constraint on PP attachment is the no-crossing rule, which prohibits lines of attachment from crossing as in (4).

(4) the man with the dog in the picture on the leash



The no-crossing rule is a strong constraint, but it can sometimes be overridden by semantic considerations. as in this example from Wendy Lehnert (personal communication):

(5) John saw the girl with the telescope in a red dress.

Attaching "the dress" to "the girl" prevents "the telescope" from attaching to the verb "saw". People are therefore forced either to attach the telescope to the girl (reluctantly, since it fits much better with "saw"), or else violate the no-crossing constraint in order to pair "girl" with "dress" and "saw" with "telescope". In the latter case they characterize the sentence as poorly worded, indicating that they are aware some grammatical constraint is not being obeyed.

A third possibility would be to attach "in a red dress" to John, but this is ruled out for cultural reasons. (Speakers prefer to violate the no-crossing constraint rather than the "no cross-dressing" constraint.)

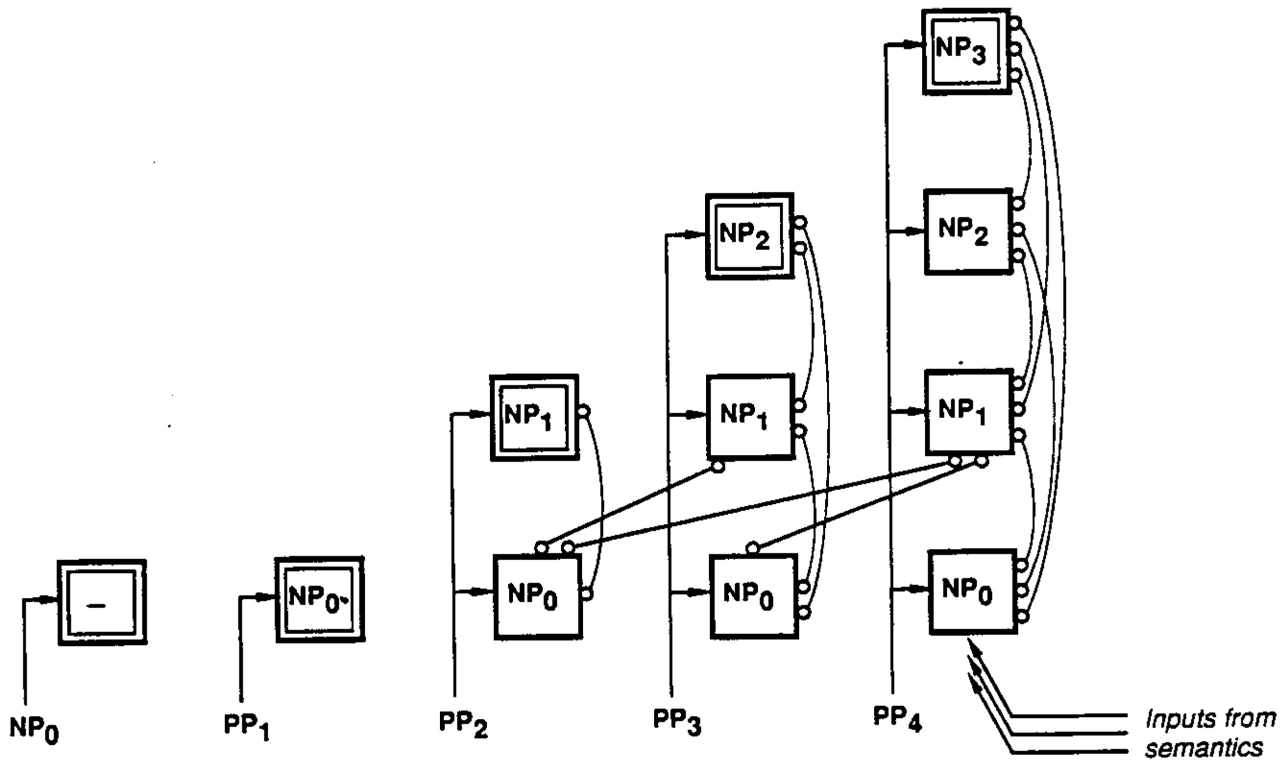


Figure 1: A connectionist network whose stable states correspond to legal PP attachments.

## 3.2 PP attachment as a competitive phenomenon

The competitive nature of the attachment problem can be captured in a simple connectionist network as shown in Figure 1. The *i*th column of units represents the alternatives for attaching PP$_i$ to any of the $i$ preceding NPs. In the figure, the presence of a PP in position $i$ feeds an excitatory stimulus to all the units in column $i$ via the vertical line with associated branches. Inhibitory connections between a column's units form a winner-take-all network that enforces the constraint that the PP may attach to only one NP. The no-crossing rule is implemented by inhibitory connections between units in different columns. A slightly lower threshold (indicated by the double line) for the topmost unit in each column produces a weak nearest-left-neighbor attachment preference. Semantic constraints supplied by other modules provide external input, either excitatory or inhibitory, as shown in the bottom right corner of the figure.

A version of this network has been simulated as an interactive activation model using equations borrowed from McClelland's Jets and Sharks model (McClelland, 1981). The network prefers simple left attachment structures such as 0-1-2, but will choose other structures if semantics warrant it. With a strong enough semantic influence it can be made to violate the no crossing rule.

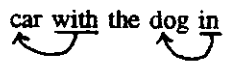
## 4 The Semantics of Attachment

The competitive attachment network handles syntactic constraints, but it says nothing about how semantic constraints are to be derived. Getting the semantics right, even for this very limited form of the attachment problem, is difficult.

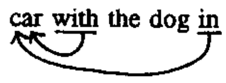
### 4.1 Naive associationism

Perhaps the simplest approach to attachment semantics is to rely on an abstract "strength of association" between pairs of NPs. Consider these examples:

(6a) the car with the dog in the cage



(6b) the car with the dog in the garage



The preferred attachment structure for (6a) is 0-1; for sentence (6b) it is 0-0. A black box that predicts the strengths of association between pairs of NPs can solve this attachment problem, as shown in Figure 2. The black box can be created by back propagation learning, If concepts are represented as feature vectors, then presumably the black box could generalize from a sufficiently large training set to handle slightly novel examples, e.g., if it learns that "dog" associates with "garage" with a strength of 0.2, it might predict a similar value for "cat".

### 4.2 Non-compositional semantics

The reason this associationist approach is deemed naive is that it completely ignores the semantics of the relationship being expressed between the two NPs. Naive associationism therefore gets the following attachment problem wrong:

(7) the car with the dog on the hood

Hoods associate far more strongly with cars than with dogs, but it's not the car that's on the hood in (7). One way to fix this is to train the black box on NP-PP pairs rather than NP-NP pairs. This requires a training set several times larger than before, since each preposition must be treated separately. The output of the black box will now be a measure of plausibility or acceptability, e.g., "car on hood" would be rated very low because it violates physical constraints, while "dog on hood" would receive a moderate rating because it is an acceptable physical state, although not a common one.

The black boxes are pattern transformers: they map NP-PP pairs into the unit interval. The competitive attachment net is even simpler, it has a small, fixed number of output states, and is basically doing categorization.

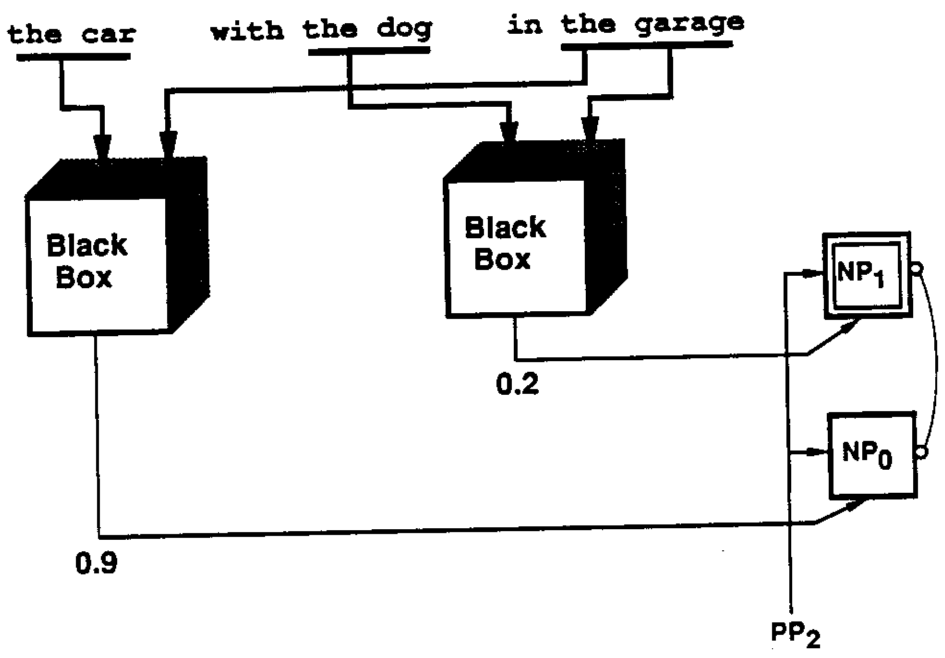


Figure 2: Solving attachment with a black box associator.

Wendy Lehnert is pursuing a variant of the black box approach in the CIRCUS system. She trains a separate black box plausibility estimator for each distinct prepositional sense. This frees the networks from having to determine the intended sense based on the two NPs, and should lead to better generalization behavior as a result.

## 4.3 Towards a compositional semantics

I call the black box approach that takes prepositions into account "non-compositional semantics," because it still ignores the fact that attaching a PP to an NP can change the meaning of the NP. This can in turn affect other attachments. In (3) for example, attaching "on the hood" to "the dog" gives the dog an implicit part-of reference to the car; this causes "with the dog on the hood" to attach to car rather than junkyard. Recognizing the part-of reference in "on the hood" is not enough; the network must transfer that property to "the dog" as a result of the attachment.

A more striking example of the effects of compositionality is (8). We can easily put the dog on the hood or in the car; a black box will happily accept either NP-PP pair in isolation. But the dog can't be in both places at once because in order to be on the car's hood he must be outside it. As soon as we actually put the dog on the hood, he can no longer go in the car.

(8) the dog on the hood in the car

In the competitive attachment net, the result of attaching "on the hood" to the dog is represented by activity in a single unit, the PP$_1$-to-NP$_0$ unit. To achieve compositionality, the attachment must result in something much more complex: the network must somehow *envision* the dog as being on the hood, and thereby in a certain physical relationship to the car, so that when it tries to put dog-on-hood inside the car it detects a spatial violation. This is the essence of compositionality: the dog composed with "on the hood" becomes a different object, with different properties than the dog in isolation.

A network with compositional semantics would require expertise in a variety of semantic primitives, such as spatial relations, part-of relations, and accompaniment, in order to know how to compose the meaning of a complex NP from the meanings of simpler NP and PP components. I am not aware of any existing connectionist architecture (other than Nature's own) that can solve this problem.

### 4.4 Further observations on attachment

Long PP chains often correspond to spatial paths that start with a local context and move to global one. For example, in (9) the reader can picture a path from the book outward to the street.

(9) the book on the shelf in the den in the house at the end of the street

The longer a string of PPs, the greater the pressure to view it as a path of some sort, which implies a simple nearest-left-neighbor attachment structure. Due to limited cognitive capacity, people are unable to handle more complex structures when the PP chains are long, just as they are unable to handle deeply center-embedded sentences. One exception to the nearest-left-neighbor convention for paths is that the last PP can optionally attach to the head NP if it describes the head rather than participates in the path. An example is (10a), which has a 0-1-2-3-4-0 attachment structure. Normally, however, one prefers (10b), whose structure is 0-0-2-3-4-5.

(10a) the book on the shelf in the den in the house at the end of the street on cockatoos

(10b) the book on cockatoos on the shelf in the den in the house at the end of the street

## 5 Relaxing the Exclusivity Constraint

Up to now we have assumed that each PP attaches to exactly one of the NP's to its left. When sentences are drawn as parse trees this is a strict requirement; otherwise the result wouldn't be a tree. But parse trees are syntactic descriptions; attachment is a semantic phenomenon. Example (3) suggests strongly that attachment relationships between NP's are more complex than syntax acknowledges. To derive the meaning of (3), the hood must participate in two relationships. The PP "on the hood" is attached to the dog, but "hood" must also attach to "car" (by an inferred part-of relation) in order to understand it as a reference to the car, indicating that the dog is on the car. Likewise, the attachment of "in the junkyard" is not exclusive. Besides describing the car, it has an implicit secondary attachment to the dog, since the dog is where the car is.

One of the important contributions connectionism can make to natural language understanding is to encourage fuzzier but richer representations. A connectionist network that could simultaneously represent all the attachment relations in a phrase, both primary and secondary, would be expressing much more of the meaning of the sentence than an exclusive attachment diagram like (3) permits. Rich, fuzzy representations are harder to program, but they are manageable if the programming is done by learning algorithms instead of human experimenters.

## 6 Discussion

There are several approaches one might take toward solving PP attachment with a more sophisticated network than a pattern transformer. One way would be to use recurrent networks to build up representations for sentences one word at a time. This idea has been pursued by St. John & McClelland (1988) for a much simpler sentence processing task, and also by Allen (1988). But attachment decisions often depend on later words, so the network would have to maintain a representation of the ambiguities in a sentence until it had seen enough of the input to settle on the correct attachment structure. Also, recurrent nets are still very simple architectures. They require huge amounts of training data to learn to recognize and exploit regularity, since they must build all their representational machinery from scratch using iterated pattern transformations. A true dynamic inferencer would start out with primitives like variable binding and distal access already in place; it would not have to construct them via extensive training.

Another approach to attachment might be to build a frame-like connectionist representation for concepts, as in DUCS (Touretzky & Geva, 1987). Attachments would be handled as slot fillers. For example, "dog on hood" would be represented by frames for dog and hood; the dog frame would have a slot containing a pointer to hood, and the slot name would include semantic features like "physical contact" and "supports from below". (In DUCS slot names are not discrete symbols; they are feature vectors.) One could build black boxes for reasoning about various primitive semantic relationships between frames, such as spatial, part-of, and support relations. These black boxes would indicate the effects of particular attachments by computing feature vectors for the slots to be added to each frame. We could then develop a competitive architecture for choosing the best attachment based on plausibility ratings. The result of making an attachment would be a modified frame which goes on to participate in further attachments, just as in a conventional symbol processing solution. The drawback to this approach is that we do not want to have to separately enumerate all possible attachments, rank order them, choose the best, and iterate; it leads to the combinatorial problems that plague conventional symbol processors.

In conclusion, I want to affirm that natural language understanding does require something like symbol processing. Connectionism makes strong, interesting suggestions about the nature of these symbols and the mechanisms that manipulate them. It offers flexible matching via parallel constraint satisfaction; fast associative retrieval; and learning algorithms that construct distributed representations and generalize to novel cases. These models also perform *compositional search*, i.e., rather than representing alternative items as separate, discrete entities (combinatorially explosive), they exist simultaneously, with overlapping representations, and they interact competitively. An example is the way rule matching and variable binding behavior emerges from simulated annealing in DCPS (Touretzky & Hinton, 1988).

We should not be discouraged by the limited ability of of weak connectionist models like categorizers and pattern transformers to handle symbol processing tasks. Nor should we abandon connectionist representations and retreat to Lisp code. Let us continue to explore the new tools connectionism has to offer, while acknowledging that ultimately dynamic inferencers, not simple pattern transformers, will be required to solve natural language.

## Acknowledgements

Can Tang implemented the interactive activation PP-attachment network. I am grateful to George Lakoff and Wendy Lehnert for their insights on the PP attachment problem, and to James McClelland, Jordan Pollack, and Paul Smolensky for helpful discussions on other matters. This work was supported by National Science Foundation grant EET-8716324, and by the Office of Naval Research under contract number N00014-86-K-0678.

## References

[1] Allen, R. B. (1987) Several studies on natual language and back-propagation. *Proceedings of the IEEE First Annual International Conference on Neural Networks*, vol. II, 287-298.

[2] Allen, R. B. (1988) Sequential connectionist networks for answering simple questions about a microworld. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pp. 489-495.

[3] Derthick, M. A. (1988) Mundane reasoning by parallel constraint satisfaction. Doctoral dissertation available as Carnegie Mellon School of Computer Science technical report number CMU-CS-88-182.

[4] Fodor, J., and Pylyshyn, Z. (1988) Connectionism and cognitive architecture. *Cognition* **28**, 3-71.

[5] Grossberg, S. (1987) Competitive learning: from interactive activation to adaptive resonance. *Cognitive Science* **11**(1), pp. 23-63.

[6] Hecht-Nielsen, R. (1987) Counter-propagation networks. *Applied Optics* **26**(23), pp. 4979-4984.

[7] Hinton, G. E. (1988) Representing part-whole hierarchies in connectionist networks. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pp. 48-54. Hillsdale, NJ: Erlbaum.

[8] 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.

[9] McClelland, J. L., & Kawamoto, A. H. (1986) Mechanisms of sentence processing: assigning roles to constituents. In J. L. McClelland and D. E. Rumelhart (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cogntion*, volume 2. Cambridge, MA: MIT Press.

[10] McClelland, J. L. (1981) Retrieving general and specific information from stored knowledge of specifics. *Proceedings of the Third Annual Conference of the Cognitive Science Society*, pp. 170-172.

[11] Newell, A. (1980) Physical symbol sytems. *Cognitive Science* **4**, pp. 135-183.

[12] Reilly, D. L., Cooper, L. N., & Elbaum, C. (1982) A neural model for category learning. *Biological Cybernetics* **45**, pp. 35-41.

[13] Rumelhart, D. E., & McClelland, J. L. (1986) On learning the past tenses of English verbs. In J. L. McClelland and D. E. Rumelhart (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cogntion*, volume 2. Cambridge, MA: MIT Press.

[14] St. John, M. F., & McClelland, J. L. (1988) Applying contextual constraints in sentence comprehension. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pp. 26-32. Hillsdale, NJ: Erlbaum.

[15] Sejnowski, T. J., & Rosenberg, C. R. (1987) Paralle networks that learn to pronounce English text. *Complex Systems* 1(1):145-168.

[16] Touretzky, 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.

[17] Touretzky, D. S. & Hinton, G. E. (1988) A distributed connectionist production system. *Cognitive Science* 12(3), pp. 423-466.

[18] Valiant, L. G. (1984) A theory of the learnable. *Communications of the ACM* 27, 1134-1142.

[19] Willshaw D. (1981) Holography, associative memory, and inductive generalization. In G. E. Hinton and J. A. Anderson (Eds.), *Parallel Models of Associative Memory*. Hillsdale, NJ: Erlbaum.