Structured Representations and Connectionist Models

Jeffrey L. Elman
Department of Cognitive Science
University of California, San Diego

ABSTRACT

Recent descriptions of connectionist models have argued that connectionist representations are unstructured, atomic, and bounded (e.g., Fodor & Pylyshyn, 1988). This paper describes results with recurrent networks and distributed representations which contest these claims. Simulation results are described which demonstrate that connectionist networks are able to learn representations which are richly structured and open-ended. These representations make use both of the high dimensional space described by hidden unit patterns, as well as trajectories through this space in time, and posses a rich structure which reflects regularities in the input. Specific proposals are advanced which address the *type/token distinction*, the *representation of hierarchical categories in language*, and the representation of *grammatical structure*.

INTRODUCTION

It seems clear that to be viable, a model of cognition should be able to represent information in a way which captures the structure of that information. Given the recent interest in connectionist models, it is natural to wonder whether such models can support structured representations of the sort that might be needed (for example) in the service of language processing.

Fodor and Pylyshyn (1988) have in fact recently argued that Classical theories, but not connectionist theories, (1) are "committed to 'complex' mental representations", and (2) have representations that reflect combinatorial structure, such that they enable structure-sensitive mental processes (p. 13). These are the principle differences. In addition, Fodor and Pylyshyn describe connectionist representations as (3) atomic, and therefore (given the limited resources available to support them) (4) finite in number (pp. 22-24).

These are strong claims. Fodor and Pylyshyn are quite right that any cognitive theory worth its salt will support complex mental representations, will reflect both the combinatorics and componentiality of thought, and will enable an open-ended number of representations. What is not self-evident is that these desiderata can only be achieved by the so-called Classical theories, or by connectionist models which are simply implementational variants. In this paper, I present results which suggest that connectionist representations can exhibit rich structure; that the representations may be complex (i.e., not atomic) and capable of reflecting both general patterns and ideosyncratic differences. Furthermore, these representations may in principle open-ended.

I begin with a brief description of the network architecture employed. I will then report results of two sets of simulations. The first explores the development of lexical categories; the second demonstrates the ability to encode syntactic information, including agreement and embedding.

ARCHITECTURE

The work which follows utilizes an architecture inspired by a model studied by Jordan (1986). Jordan demonstrated the utility of allowing recurrent connections from output units. In the form of the network I have been studying, shown in Figure 1, in addition to the usual input units, output units, and hidden units, there are a set of context units which hold a copy of the hidden unit activations (on a one-to-one basis) from the prior cycle. These context units then feed back into the hidden units (on a fully distributed basis) on the next cycle. The hidden units have the task of mapping the input to the output, and because the input now includes their own prior states, they must

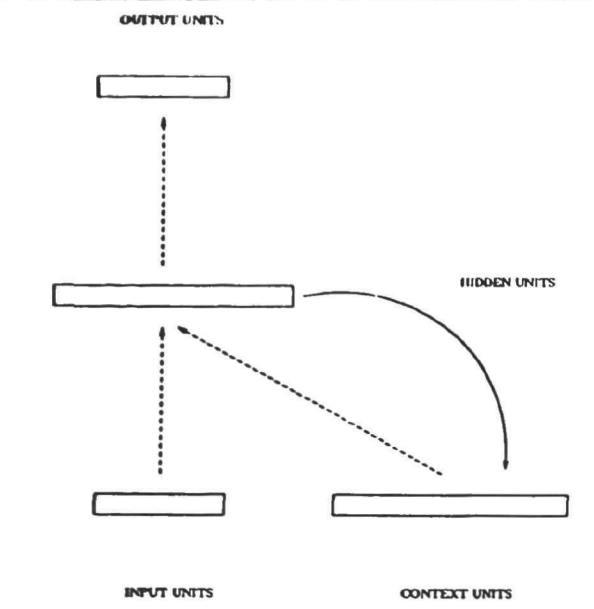


Figure 1.
Network with recurrent connections from hidden units to context units.

develop representations which serve as memory as well. Note that this approach to memory relies on distributed rather than localist representations. Memory is not associated with individual nodes but rather with the state vector on the context units. Furthermore, this notion of memory is highly task specific.

This architecture, which I will call a Simple Recurrent Network (SRN) has been studied in Elman (1988, 1989); Hare, Corina, & Cottrell (1988); and Servan-Schreiber, Cleeremans, & McClelland (1988), and will be used for the two simulations I report here. It is particularly relevent in the domain of language, since it allows for the processing of serial inputs. Thus, language can be processed naturally on an element-by-element basis.

DISCOVERING LEXICAL CATEGORIES

One area of language which exhibits rich structure is lexical categorization. Lexical categorization is manifested in a number of ways; in English, one of these manifestations is word order. Not all classes of words may appear in any position. Furthermore, certain classes of words, e.g, transitive verbs, tend to co-occur with other words, e.g., nouns as direct objects (although as will be relevant in the next simulation, the co-occurrence facts may be complex).

The goal of the first simulation was to see if a network could learn the lexical category structure which is implicit in a language corpus. The overt form of the lexical items was arbitrary; however, the behavior of the lexical items — defined as their co-occurrence restrictions — reflected their membership in implicit classes and subclasses. The question was whether the network could induce these classes.

Stimuli, Task, and Network

A lexicon of 29 nouns and verbs was chosen. Words were represented as 31-bit binary vectors (two extra bits were reserved for another purpose); each word was randomly assigned a unique vector in which only one bit was turned on. A sentence-generating program was then used to create a corpus of 10,000 2- and 3-word sentences. The sentences reflected certain properties of the words; for example, only animate nouns occurred as the subject of the verb eat. Finally, the words in successive sentences were concatenated, so that a stream of 27,354 vectors was created. This was the input set.

The task was simply for the network to take successive words from the input stream and to predict the subsequent word (by producing it on the output layer). After each word was input, the output was compared with the actual next word, and the backpropagation of error algorithm (Rumelhart, Hinton, & Williams, 1986) was used to adjust weights. Words were presented in order, with no breaks between sentences. The network was trained on 6 passes through the corpus.

Results

Because the sequence is non-deterministic, short of memorizing the sequence, the network cannot succeed in exact predictions. Nonetheless, the network does learn to approximate the expected frequency of occurrence of successor words. The rms error, using the empirically derived

probability of occurrence of successors, was 0.053; the cosine of the angle between output vector and likelihood vectors (which normalizes for length differences) was 0.916, indicating a close match.

Discussion

I would like to focus on *how* the network accomplishes the task. One way to do this is to see what sorts of internal representations the network develops in the process of trying to carry out the prediction task. These representations are captured by the pattern of hidden unit activations which are evoked in response to each word in its context. These patterns were saved during a testing phase, and then subjected to hierarchical clustering analysis. Figure 2 shows the tree constructed from the hidden unit patterns for the 30 lexical items, where each item is the average of for a word across all the contexts in which it occurrs in the testing data.

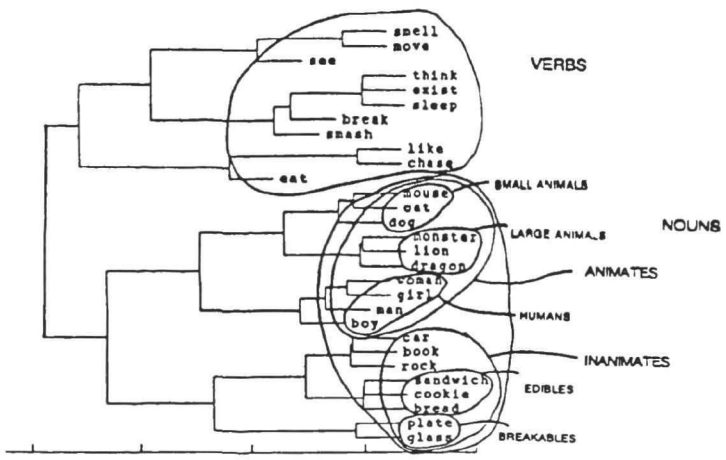


Figure 2.
Hierarchical cluster analysis of the average hidden unit activation patterns for each of the 29 unique words in the word-prediction simulation.

The network has discovered that there are several major categories of words. One large category corresponds to *verbs*; another category corresponds to *nouns*. The verb category is broken down into groups which take animate subjects; which are intransitive or take optional objects, and which require direct objects. The noun category breaks down into major groups for *inanimates* and *animates*; the animates are divided into *large animals* and *small animals*, and *humans*. Inanimates are divided into *breakables*, *edibles*, and miscellaneous.

This category structure reflects facts about the possible sequential ordering of the inputs. The network is not able to predict the precise order of words, but it recognizes that (in this corpus) there is a class of inputs (viz., verbs) which typically follow other inputs (viz., nouns). This knowledge of class behavior is quite detailed; from the fact that there is a class of items which always precedes chase, break, smash, it infers a category we might call *aggressors*.

Several points should be emphasized. First, the category structure is hierarchical. The hierarchicality is achieved through the organization of the representational space described by hidden unit patterns, with higher-level categories corresponding to larger and more general regions of space. Second, the categories are not discrete. Category boundaries are smooth, and category membership may be marginal or ambiguous (although it may also be clear and unambiguous). Finally, the content of the categories is not known to the network. The network has no information available which would "ground" the structural information in the real world. In this respect, the simulation has much less information to work with than is available to real language learners.

Types and tokens. The tree shown in Figure 2 was constructed of activation patterns averaged across context. When the context-sensitive hidden unit patterns are clustered, it is found that the large-scale structure of the tree is identical to that shown in Figure 2. However, each terminal branch now continues with further arborization for all occurrences of the word (no

instance of any lexical item appears in the branch of another.

This is an important finding. It verifies that the representation of each lexical item, wherever it occurs, reflects the constraints to which the item is subject *as a lexical type*. The representations clearly mark typehood. But the patterns also individuate the different tokens of types. No two tokens are precisely identical. They are different because they have occurred in different contexts, and the representations are *highly context-sensitive*.

Even more interesting is that there is a fine sub-structure to the various tokens of a type. For instance, tokens of **boy** which occur in subject position tend to cluster together, and apart from tokens of **boy** in object position. The same pattern occurs among the representations of tokens of other nouns. This detailed sub-grouping makes it possible for the network to distinguish *tokens* of a *type*, as well as different *types*. Usefully, the tokens are themselves organized in a manner which reflects systematic facts about the context in which they occur.

REPRESENTATION OF SYNTACTIC STRUCTURE

In the previous simulation there was little interesting grammatical structure. Sentences were short and simple and most of the patterning was explained at the level of properties of individual lexical items. In the next simulation we develop representations which reflect more complex syntactic structure. A phrase structure grammar, shown in Table 1, was used to generate training corpora. Each word was represented by a localist 26-bit vector in which each bit stood for a different word. Training proceded incrementally. A network similar to that shown in Figure 1 was trained on the prediction task. The training data consisted of an initial set of 10,000 sentence corpus of simple sentences; the percentage of complex sentences was gradually altered over the course of training from 0% to 75%. Mean sentence length of the final training set was 5.3 words (range: 3 to 13 words). This simulation superficially resembles the previous one, except that the sentences were more complex and reflected a variety of syntactic constraints. Specifically,

---

S -> NP VP "."
NP -> PropN | N | N RC
VP -> V (NP)
RC -> *who* NP VP | *who* VP (NP)
N -> *boy* | *girl* | *cat* | *dog* |
boys | *girls* | *cats* | *dogs*
V -> *hit* | *feed* | *see* | *hear* | *walk* | *live* |
hits | *feeds* | *sees* | *hears* | *walks* | *lives*

Additional restrictions:

- number agreement between N and V within clause, and (where appropriate) between head N and subordinate V
- verb arguments:
  hit, feed —> require a DO
  see, hear —> optionally take DO
  walk, live —> preclude a DO
  (observed also for head/verb relations in relative clauses)

Table 1

---

it was necessary that the network to learn the following:

- Agreement. Subject nouns agree in number with their verbs.
- Verb arguments. One class of verbs requires a direct object; a second class optionally permits a direct object; and a third class never occurs with a direct object.
- Relative clauses. The presence of relative clauses requires that the network maintain agreement and verb argument relations within the appropriate clause, and despite the presence of intervening clausal material. In **dogs who boy feeds see cat**, agreement occurs between N1 and V2, and between N2 and V1. Similarly, because this sentence involves an object-headed relative clause, the network is required to learn that although the verb **feeds** normally is followed by a direct object, that position has already been filled by the prior word **dogs**.
- Sentence completion. The network is required to develop a sense of what are candidates for complete grammatical sentences, by predicting when a sentence ending (".") may occur.

At the conclusion of training, network performance was measured by comparing the

outputs with the empirically derived conditional probability of occurance for each possible word; the mean cosine of the angle between the vectors was 0.92 (SD: 0.19). Network predictions in various contexts are illustrated in Figure 3. As can be seen, the network succeeds in predicting the class of words which appropriately follows in each context. This is true even in complex sentences where relative clauses render useless any generalization based on the linear order of words in simple sentences.

Again, we may ask how the network has achieved this performance. For these purposes, it is important to be able to look at the time-varying states of the network as it processes various sentence types. This information is not easily revealed in hierarchical clustering, so the following procedure involving principal component analysis was developed.

The final training set was passed through the network a final time, and hidden unit patterns were saved. The covariance matrix of these vectors was calculated; the eigenvectors of this matrix were then used as the basis for describing hidden unit vectors. This basis tends to provide a somewhat more interpretable (and localist) view of the hidden units' distributed representations. Furthermore, the dimension are ordered (using the eigenvalues) by decreasing importance in accounting for variability. Thus, we can chose to look at only a few of the dimensions (the principal components, or PC's) and plot the movement through this reduced space as the network processes sentences of interest.

Figure 4 displays state trajectories which illustrate the representation of verb-argument structure. Trajectectories through PC 1x3 space are shown for the three sentence fragments **boy hits** .... (**hits** requires a direct object), **boy sees** ...., (**sees** optionally takes a direct object) and **boy walks** .... (**walks** never occurs with a direct object). The initial state after processing the first word is the same for all three sentences. However, there is a systematic displacement in PC 1x3 space which corresponds to the expectation of a direct object. This pattern holds true over a wide variety of contexts and more complex syntactic structures. Figure 5 shows the manner in which embedding is represented. There is a basic trajectory in PC 1x11 space which is associated with simple sentences; this trajectory is replicated and shifted in space to indicate subordinate clauses.

CONCLUSIONS

Several things may be said about the results of these simulations.

First, it is quite apparent that connectionist representations may be quite rich. They need not be atomic, but may instead possess structure which reflects the systematic patterns which are immanent in the primary data. The representational structure is embodied in the state of the network; different states are associated with different syntactic structures.

Second, the representations in these simulations are highly context-sensitive. This sensitivity co-exists comfortably with the ability to capture systematic patterns which are more generally true. Indeed, the same mechanism is responsible for both aspects of representation. As a consequence, connectionist representations bind the semantics of reference with the syntax of representation. Classical theories have long grappled -- with less than satisfactory results, in the view of many -- with the tension that is produced when one insists that syntactic and semantic representations be kept distinct and that there be no direct interaction between them. It is a natural consequence of connectionist representations that there be a common language which simultaneously supports syntax and semantics. This suggests the distinctions between the syntax and semantics may be quantitative in nature and do not stem from any deep distinctions.

Third, we have seen how the distributed representations which are developed at the hidden unit level make use of state space and state dynamics. The representational space is organized to reflect the structure that is implicit in the primary data. The further dimension of time, captured in the notion of recurring trajectories through space, adds to the representational power and permits statements to be made about syntagmatic relations. Furthermore, we have found that

although the representations are highly distributed, this does not mean that they are unanalyzable. It is possibly to insightfully characterize the organization of the representational space, and to discover regularities in patterns of temporal movement through that space.

Fourth, the representational system has itself been inferred from the data. The architecture ultimately *limits* the representational power of a network, of course, but it does not *specify* the representational form. We do not begin with notions of lexical categories or grammatical patterns. These concepts are present in the data and are learned by the network. These results of course do not deny the importance of evolutionary mechanisms in constraining the mechanisms which support language processing. The findings here simply suggest that a simple but powerful learning algorithm such as backpropagation is capable of extracting rather more information from raw input than one might have supposed.

Fifth, the representational system is relatively open-ended. Just as it was not necessary to stipulate categories or structures prior to learning, it was not necessary to place an upper limits on the number of categories or depth of structure.

We have seen how these characteristics can lead to solutions of the *type/token* distinction, to the discovery and representation of *lexical categories*, and to the representation of certain aspects of *syntactic structure*. These are important problems in language theory, and the approach suggested here is highly encouraging. However, language presents many problems of a highly complex nature, and the simple successes obtained here should not cause us to forget just how difficult those problems can be.

There are also serious limitations to the work here which must be pointed out. The most basic has to do with the nature of the task. The prediction task has been useful in these simulations. The appeal of this task is that it makes minimal assumptions about prior knowledge on the part of the learner (or network). But while prediction or anticipation may be a plausible activity during language comprehension, it can hardly be the primary basis for learning language.

It is not clear what an appropriate task is, but there have been recent suggestions which go in the right direction. St. John and McClelland (1988) have described a system in which sentence inputs are used to construct an interpretation of events in the world. This is a far more plausible view of what is involved in processing language than suggested by the prediction task. One question that remains unanswered in the St. John and McClelland model is where the primitive notions such as *patient* and *agent* come from (these figure importantly as built-in constructs in their network), and how to extend the task to complex sentences. Strikingly, these are just the sorts of questions which are addressed by the present approach. It is appealing to think that the two approaches might be combined.

In conclusion, the simulations described here explore a new approach to the representation of language. While there are many deep and important questions to be answered, this approach provides a glimpse of the sort of representational power which a connectionist theory of language could have.

ACKNOWLEDGEMENTS

I would like to thank Jay McClelland, David Rumelhart, and Mary Hare for many useful discussions. This work was supported by contract N00014-85-K-0076 from the Office of Naval Research and contract DAAB-07-87-C-H027 from Army Avionics, Ft. Monmouth.

REFERENCES

Elman, J.L. (1988). Finding structure in time. CRL Technical Report 8801. Center for Research in Language, University of California, San Diego.

Elman, J.L. (1989). Structured representations and connectionist models. CRL Technical Report 8901. Center for Research in Language, University of California, San Diego.

Fodor, J., & Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A critical analysis. In S. Pinker & J. Mehler (Eds.), *Connections and Symbols* (pp. 3-71). Cambridge, Mass.: MIT Press.

Hare, M., Corina, D., & Cottrell, G. (1988) Connectionist perspective on prosodic structure. CRL Newsletter, Vol. 3, No. 2. Center for Research in Language, University of California, San Diego. D.E. Rumelhart & J.L. McClelland (Eds.). (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Volume II: Psychological and biological models* Cambridge, Mass.: MIT Press.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation. In D.E. Rumelhart & J.L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1) (pp. 318-362). Cambridge, Mass.: MIT Press. D.E. Rumelhart & J.L. McClelland (Eds.). (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Volume I: Foundations* Cambridge, Mass.: MIT Press.

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

St. John, M., & McClelland, J.L. (1988). Learning and applying contextual constraints in sentence comprehension. Technical Report. Department of Psychology. Carnegie-Mellon University.

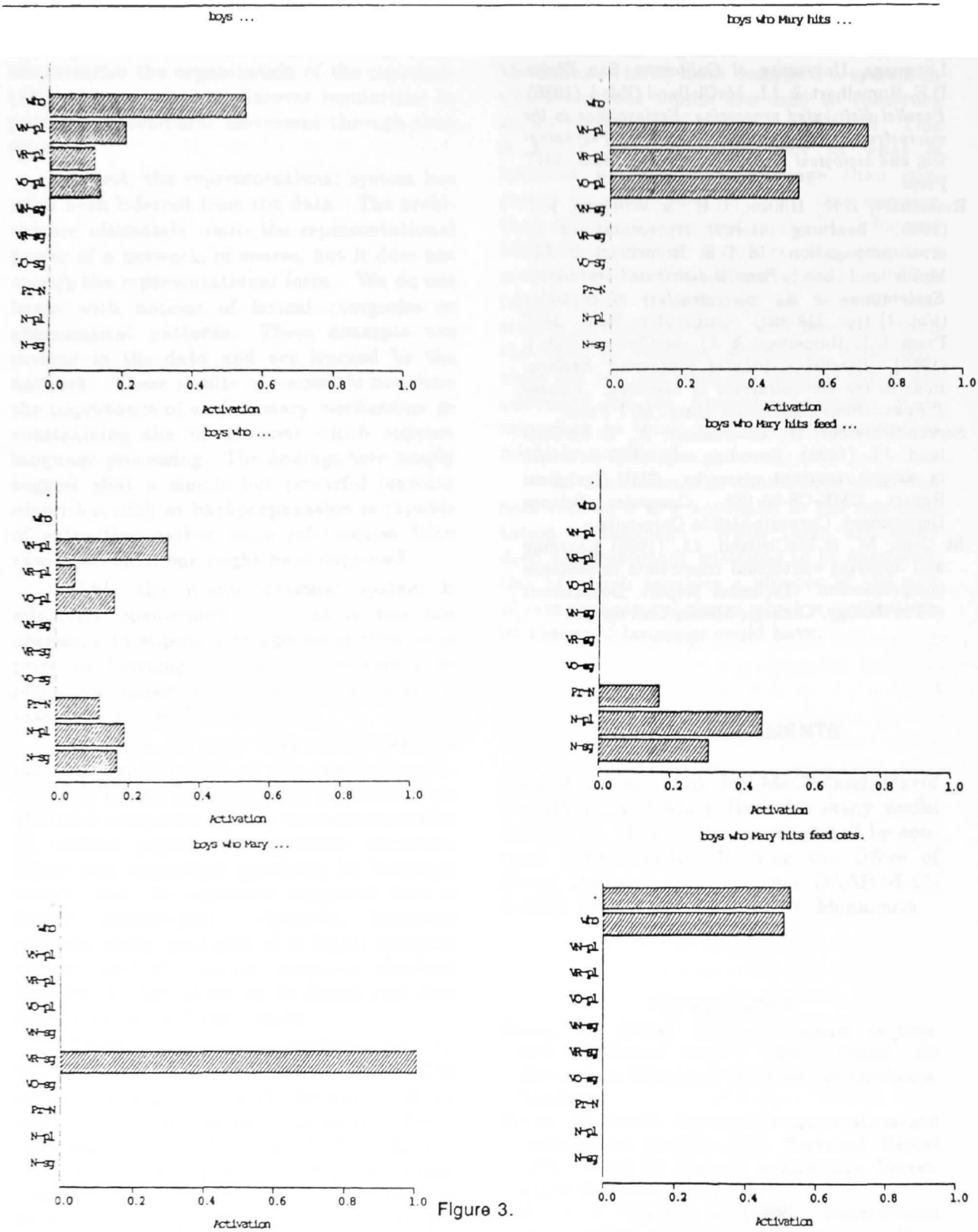
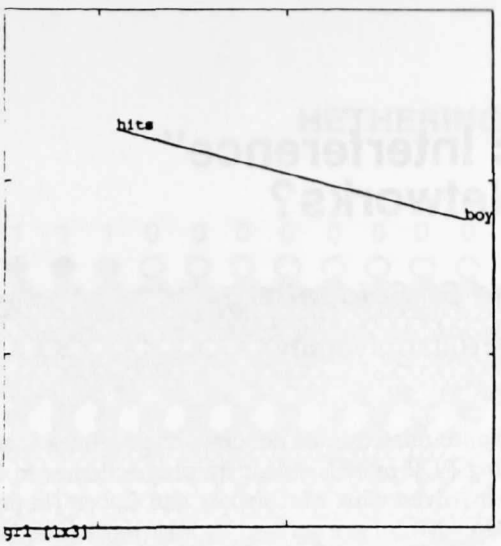
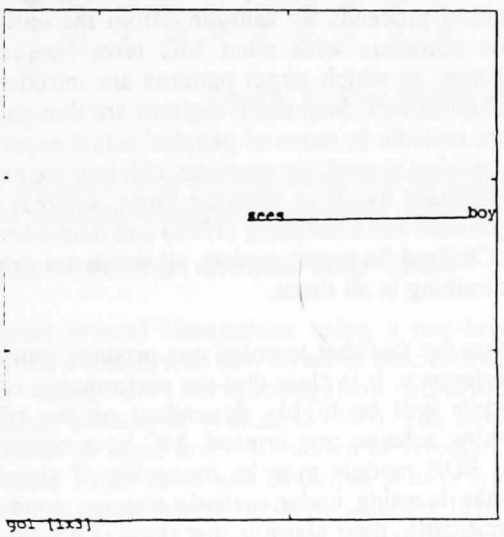

boys ...
boys who ...
boys who Mary ...
boys who Mary hits ...
boys who Mary hits feed ...
boys who Mary hits feed cats.
Activation
0.0
0.2
0.4
0.6
0.8
1.0

Figure 3.





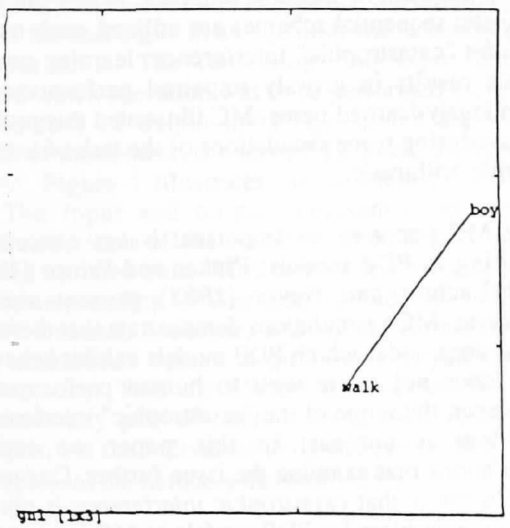
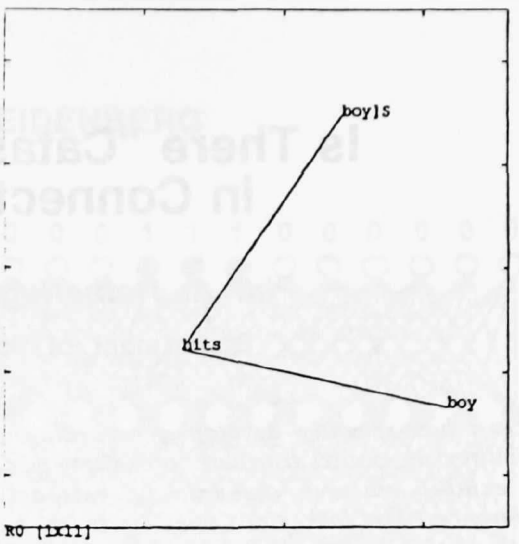
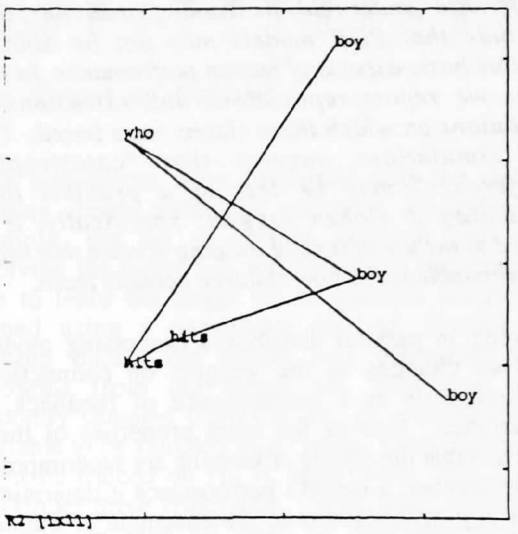


Figure 4.





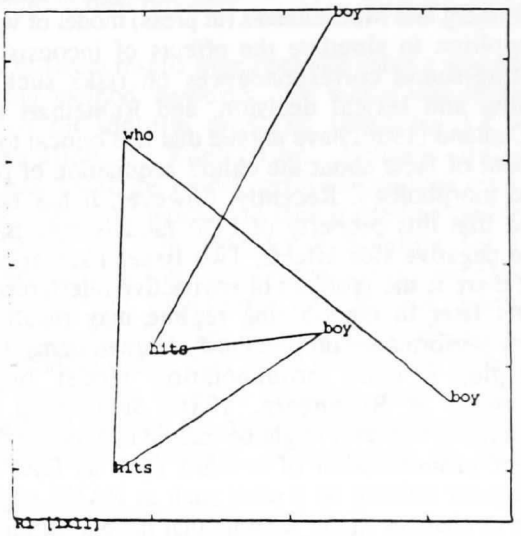


Figure 5.