Dynamic Inferencing in Parallel Distributed Semantic Networks*

Ronald A. Sumida
Artificial Intelligence Laboratory
Computer Science Department
University of California
Los Angeles, CA, 90024

Abstract

The traditional approach to dynamic inferencing is to represent knowledge in a symbolic hierarchy, find the most specific information in the hierarchy that relates to the input, and apply the attached inferences. This approach provides for inheritance and parallel retrieval but at the expense of very complex learning and access mechanisms. Parallel Distributed Processing (PDP) systems have recently emerged as an alternative. PDP systems use a very simple processing mechanism, but can only access high-level knowledge sequentially and require an enormous amount of training time. This paper presents Parallel Distributed Semantic (PDS) Networks, an approach that integrates the best features of symbolic and PDP systems by storing the content of symbolic hierarchies in ensembles of PDP networks, connecting the networks in the manner of a semantic network, and using Propagation Filters to determine how information is passed between networks. Simulation results are presented which indicate that PDS Networks and Propagation Filters are able to perform pattern completion from partial input, generate dynamic inferences, and propagate role bindings.

Introduction

In order to dynamically generate inferences, a natural language understanding system must be able to access knowledge structures at varying levels of generality and to use the associated information in making the proper inferences. For example, consider the following texts which involve both general and specific information about various hit actions.

Text 1
John hit Mary because she dumped him for Bill.

Understanding why John hit Mary involves accessing general information about why humans hit one another, whereas understanding:

Text 2
Douglas hit Tyson. He wanted to win the title.

requires utilizing the specific knowledge that one reason why boxers hit one another is to win a competitive activity.

Previous Work

Symbolic/Localist systems represent general and specific knowledge by forming an explicit hierarchy with each concept represented by a symbol or node, and with related inferences attached at the appropriate level. Figure 1 is an example of a semantic network with part of a simplified hierarchy for understanding the texts above. At the top of the hierarchy is the general hit act (HIT), followed by levels for one person hitting a person (HUMAN-HIT-HUMAN), and a boxer hitting his opponent (BOXER-HIT-BOXER). Each of these levels is connected to knowledge about the motivation for the hit act.

The language understander generates inferences by: (1) searching the hierarchy for the most specific node that applies to the input and binding the appropriate roles, (2) applying the attached inferences, and (3) propagating the role bindings. A number of methods for searching the hierarchy have been proposed, such as localist connectionist spreading activation (e.g. [Waltz and Pollack, 1985],[Sumida *et al.*, 1988]).

The advantage of the hierarchical approach is that it provides an economical method for representing structured knowledge (through inheritance) and it allows large amounts of knowledge to be searched in parallel (by applying the search procedure simultaneously to different parts of the network). Unfortunately, this approach suffers from three major shortcomings. First, it is not clear how specific the information represented in the hierarchy should be. Should there be an explicit level for a male hitting a female? Second, a combinatorial explosion problem occurs when new information is added to memory, since there are an enormous number of ways that shared features can be combined to index the new information. Third, the process of searching the hierarchy introduces serious complexities. With spreading activation systems, for example, the search

*This research is supported by a grant from the Keck Foundation.

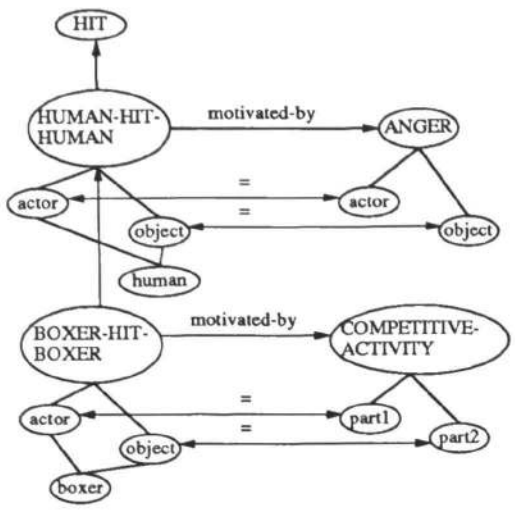

Figure 1: Semantic network with a hierarchy of HIT acts. Single arrows indicate parent/child (is-a) relationships. Straight lines indicate role/filler relationships. Double arrows labeled "=" signify equivalence relationships.

procedure itself is simple but the complexity is hidden in the procedure for determining the link weights.

Parallel Distributed Processing (PDP) Systems represent concepts as patterns of activation in a highly interconnected network of units [Rumelhart and McClelland, 1986]. PDP systems provide inheritance between concepts by having the patterns for a general concept and its descendants share parts, so that any effects caused by the pattern for the general concept will be transferred over to its descendants [Hinton, 1981]. Exceptions are encoded by the parts of the pattern that are not shared, so that the unique parts of the pattern can override information from the ancestors. Propositional information is stored in PDP systems by using triples of the form (Role1 Relation Role2). A sequence of propositions can be encoded in a PDP network (e.g. [Pollack, 1988]) by generating a reduced description (over the hidden units) for a triple, and presenting it as input along with the next triple in the sequence.

PDP systems offer a solution to the problems plaguing symbolic/localist systems for storing hierarchical information and generating inferences because they: (1) do not need to make a discrete decision about what knowledge structures to include in the network. The extent to which a concept exists in the hierarchy is determined by statistical correlations between training patterns, (2) avoid combinatoric problems since the network automatically determines (through the learning procedure) which of the shared features are salient, and (3) use an extremely simple processing mechanism.

Unfortunately, PDP systems suffer from their own unique problems because they have a very limited representation of structured knowledge and they store too much information in a single network. The first problem is that it takes an enormous amount of time to train the network because of the incredibly large number of concepts that it must hold. The second problem is the *Knowledge-Level Parallelism Problem* [Sumida and Dyer, 1989]. Since the network can only store or retrieve one triple at a time, it is not possible to search large amounts of knowledge in parallel. This is particularly problematic in inferencing, where a large number of alternatives must be pursued in parallel for efficiency reasons.

Parallel Distributed Semantic (PDS) Networks

The PDS network approach is to combine the best features of symbolic/localist semantic networks and PDP approaches by: (1) Storing the content of a symbolic hierarchy in an ensemble of PDP units. For example, the HIT hierarchy of Figure 1 is stored in a PDP network, as shown in Figure 2. Similarly, information about humans is stored in a second PDP network. Each level of the hierarchy is represented as a pattern of activity over the appropriate ensemble. For instance, HUMAN-HIT-HUMAN and BOXER-HIT-BOXER are stored as alternative patterns of activation over the HIT ensemble. (2) Connecting the PDP networks according to the structure of a semantic network. In Figure 2, the HIT and HUMAN PDP networks are connected in the same general manner as in Figure 1.

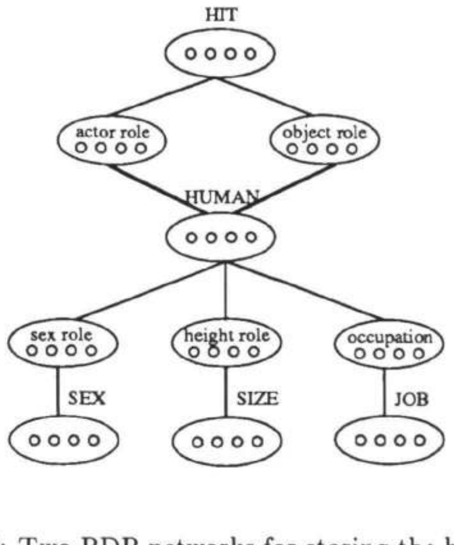

Figure 2: Two PDP networks for storing the hierarchy of Figure 1. The circled dots in the figure indicate an ensemble of units. The conceptual ensembles (i.e., HIT and HUMAN) are labelled with capital letters and the role ensembles are indicated by lower-case labels placed inside the oval of units.

In a previous paper [Sumida and Dyer, 1989], we showed how PDS networks store *static* role bindings, in which a previously encountered concept is bound to a role of a *known* proposition. This paper demonstrates how PDS networks have been expanded to create generalizations and to generate *dynamic* inferences. In order to generate dynamic inferences, PDS networks: (1) store each hierarchy in the appropriate network by presenting training instances and having the network automatically generate the proper generalizations, (2) classify new input according to the proper level of the hierarchy from the generalizations made during training, and (3) propagate patterns for the role bindings to the proper, related networks. For example, to understand Text 2 using PDS networks: (1) the HIT hierarchy is stored by training the HIT network on a number of instances of humans hitting one another and boxers hitting boxers. The network then generalizes the training data to learn patterns for HUMAN-HIT-HUMAN and BOXER-HIT-BOXER, (2) a new pattern, Douglas hit Tyson, is presented and classified as the BOXER-HIT-BOXER level of the hierarchy that was generalized from the training instances, and (3) the Douglas and Tyson patterns are propagated to the COMPETITIVE-ACTIVITY network.

Storing Information in PDS Networks

Hierarchical information is stored in a PDS network by generating a unique pattern (for each training instance) that represents a *reduced description* of the input. The network automatically generalizes from these training patterns to generate a pattern that represents a level of the hierarchy. For example, to represent the BOXER-HIT-BOXER level from Figure 1, a number of instances of boxers hitting one another are presented to the network, a unique pattern is generated for each, and the network generalizes from these patterns to generate the pattern for BOXER-HIT-BOXER. A reduced description of the input is generated by using a slightly modified version of the standard encoder network. In a standard encoder network, the input and output layers have the same number of units and are presented with exactly same pattern. The weights are modified so that the input patterns are trained to recreate themselves as output. The resulting pattern over the hidden units is taken to represent the reduced description. In the modified network that we use, the same set of units is used for both the input and output layers. The network can be viewed as an encoder net with the output layer folded back onto the input layer and with two sets of connections: from the single input/output layer to the hidden layer and from the hidden layer back to the input/output layer.

As an example, consider the simplified figure of the HUMAN ensemble shown in Figure 3. The network relates the representation of a human with the features of that person. The sex, height, and occupation role groups cumulatively represent the input/output layer and the HUMAN group represents the hidden units. The black arrows indicate connections from the input/output layer to the hidden layer, and the grey arrows indicate connections from the hidden layer back to the input/output layer. The thick lines in the figure connect a role group to an ensemble which can fill it and indicate links which propagate a pattern without changing it. The jagged lines in the figure are suggestive of the patterns of activation that are presented to the network. Suppose that we want to train the network of Figure 3 on a tall, male boxer. We need to generate a unique pattern in the HUMAN ensemble that is a reduced description of the input and that represents the boxer. The pattern is generated by placing the features for the boxer into the appropriate ensembles (male into the SEX ensemble, tall into the SIZE ensemble, and boxer into the JOB ensemble), propagating the patterns into the input/output role ensembles (male from SEX to sex role, tall from SIZE to height role, and boxer from JOB to occupation role, as indicated by the single dotted arrows in the figure), and training the network by altering the connections between the role groups and the HUMAN units that form the hidden layer (the double dotted arrows indicate the training process). The resulting pattern over the HUMAN units (Boxer1) represents the boxer.

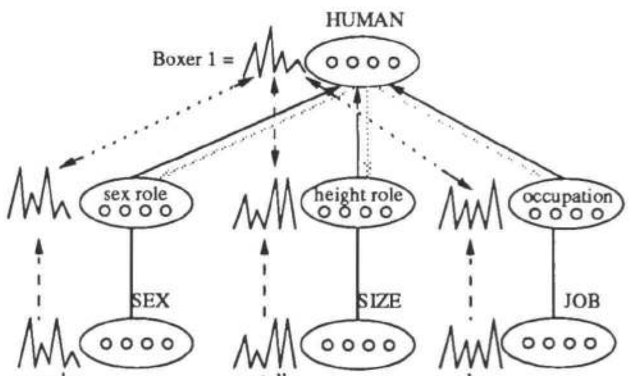

Figure 3: Representing a boxer in the HUMAN network.

Suppose that we now want to represent a proposition for one boxer hitting another, specifically, (Boxer1-HIT-Boxer2). The idea is to use exactly the same method described above for representing a human, but this time applied to the HIT network of Figure 4. Thus, the Boxer1 pattern is propagated from HUMAN to the actor role of HIT, the Boxer2 pattern is generated in the same manner described above and then propagated from HUMAN to the object role of HIT, and the network is trained so that the input/output pattern (the conjunction of Boxer1 and Boxer2) will recreate itself as output. The resulting pattern over the hidden HIT units is the representation and reduced description for the proposition (BOXER1-HIT-BOXER2).

The other training instances are created in exactly

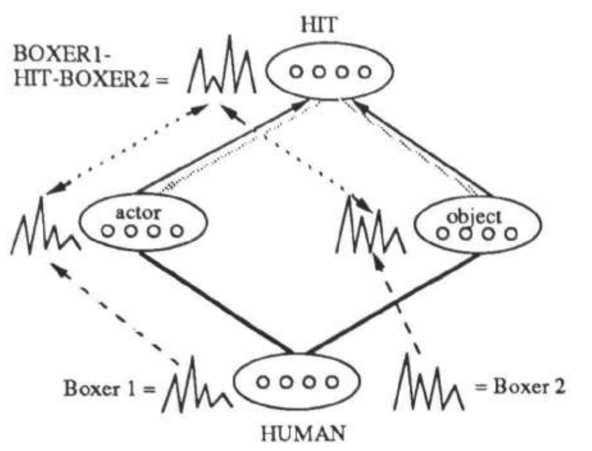

Figure 4: Representing one boxer hitting another in the HIT network.

the same manner. Various humans are represented in the HUMAN network by presenting their features and generating a reduced description. Various hit acts are represented in the HIT network by presenting the actor and object and again generating a reduced description.

Classifying the Input and Propagating Role Bindings

As a result of the training process, the hidden units learn to classify the input by responding to common features of the training patterns. Particular hidden units develop patterns that generalize from the features shown during training and thus represent a level of the hierarchy. For example, in the simulations described in the next section, two of the HIT units respond with a pattern of "01" when two boxer patterns are presented. The hidden units allow the network to classify a new concept based on its similarity to ones seen during training. When the pattern for the new concept is presented, the hidden units that are sensitive to its features become activated. The pattern over these units serves to classify the input. The hidden units also help the network perform pattern completion from partial or noisy input, so that when an unfamiliar pattern is presented that is similar to a known one, the hidden units will respond to the similar features and recreate the known pattern.

The subset of hidden units that classifies the input acts as a *Propagation Filter* that directs where role bindings are propagated. Propagation filters are based on the idea of skeleton filters [Sejnowski, 1981, Hinton, 1981] which use a pattern over one group of units to enable a restricted subset of a group of filter units. Propagation filters consist of: (1) groups of filter units, each of which gate the connection from a source to a destination and (2) a set of selector units that choose which filter group to enable. The pattern over the selector units opens up the proper filter group by driving its units above threshold. For example, in Figure 5, the "01" pattern over units 2 and 3 of the selector opens up group1 of the filter, which allows the pattern to be passed from source1 to destination1. The pattern in source2 is not propagated because the units in group2 remain well below threshold.

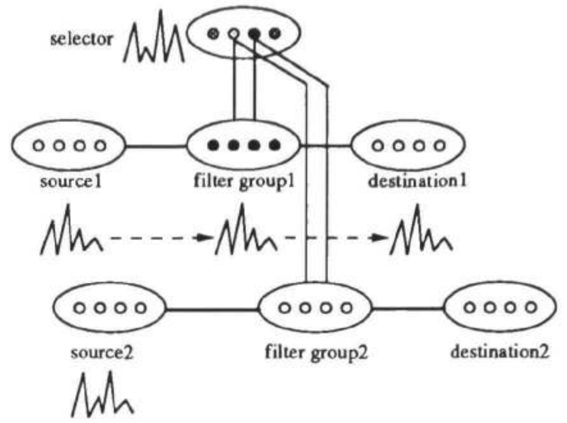

Figure 5: A Propagation Filter with group1 enabled and group2 disabled.

To illustrate how propagation filters are applied to propagating role bindings, suppose that we train the network on numerous examples of boxers hitting one another and of people hitting other people. As we mentioned earlier, one result of the training is that two of the HIT units learn to respond with "01" when two boxer patterns are presented and with "10" or "00" when two non-boxers are presented. These two units constitute a selector that enables the filters connecting the roles of HIT with the roles of COMPETITIVE-ACTIVITY, provided that the pattern over the two units is "01". Similarly, the two units enable the filters connecting the roles of HIT with the roles of ANGER if the pattern is "10" or "00". For example, suppose that we test the network on [Douglas HIT Tyson] by generating the pattern for Douglas in the HUMAN network (by the procedure shown in Figure 3), propagating the Douglas pattern to the actor role of HIT, and generating the Tyson pattern in HUMAN and propagating it to the object role of HIT. As illustrated in Figure 6, the network classifies it as BOXER-HIT-BOXER by generating the "01" pattern over the two HIT units. The "01" pattern enables the filter group that connects the actor of HIT with the first participant role of COMPETITIVE-ACTIVITY and the object of HIT with the second participant role of COMPETITIVE-ACTIVITY. Thus, the Douglas pattern is propagated to the participant1 role and Tyson to participant2. The filters that connect the actor and object roles of HIT with the roles of ANGER (not shown in the figure) are not enabled, since they require that the two selector units have a pattern of "10" or "00". As a result, the Douglas and Tyson patterns are not propagated to ANGER. Note that if the input had been [John HIT Mary] as in Text 1, then the two HIT units would respond with "10", which would instead open the filters from HIT to ANGER and result in John being propagated to the actor role of ANGER and Mary to the object role.

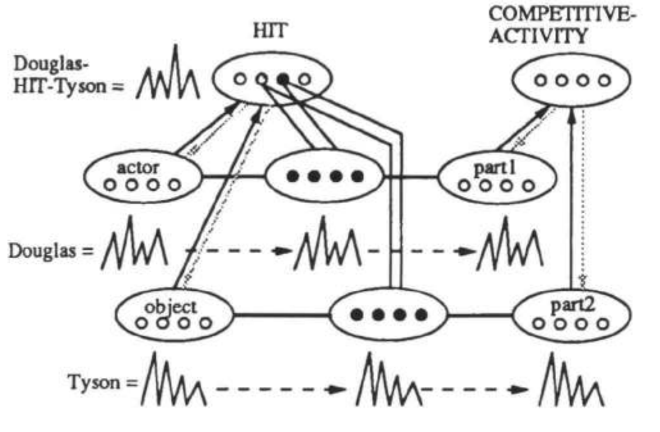

Figure 6: Testing the HIT network on [Douglas HIT Tyson]. The "01" pattern over the two units of HIT opens the filters connecting HIT with COMPETITIVE-ACTIVITY, and allows the Douglas and Tyson patterns to be propagated.

Implementation Details and Simulation Results

PDS Networks are implemented in a natural language understanding system called DCAIN which is written in the C programming language and is fully implemented on the examples shown above. In our simulations, the HUMAN network was divided into the following 8 role groups: occupation (boxer, businessman, career woman or teacher), strength (strong, average, or weak), height (tall, medium, or short), weight (heavy, average, or light), sex (male or female), hair color (blond, brown, red, or black) and length (short, medium, or long), eye-color (blue or brown), and skin-color (dark or light).

The following is a list of the tasks on which PDS networks were tested. In every case, the network performed successfully: (1) Retrieving the correct (i.e., consistent with the training data) features given a single characteristic. For example, given the boxer occupation, the network correctly filled in the strong, tall and heavy features. (2) Retrieving the correct human given enough distinguishing characteristics. (3) Categorizing familiar HIT acts (i.e., those seen during training) and generating the proper inferences. (4) Categorizing *new* HIT acts involving familiar humans and generating the proper inferences. New hit acts were created by selecting a pair of familiar humans whom the network had not seen hit one another. (5) Categorizing new HIT acts involving new humans and generating the proper inferences. New humans were created by varying irrelevant features (such as hair and eye-color) of the humans that the network had seen during training.

Future Work

DCAIN needs to be expanded to: (1) dynamically form the overall PDS network structure. DCAIN can add new information and form generalizations within a *single* network while avoiding the problems of localist systems. However, DCAIN does not currently learn the overall structure of the system, and must be expanded to do so. (2) include structural language information and incorporate more conceptual knowledge, (3) address the issue of ambiguity, (4) address timing and sequencing issues, and (5) incorporate elements of explanation-based generalization.

Conclusions

Parallel Distributed Semantic (PDS) Networks store the content of symbolic hierarchies in ensembles of PDP networks, connect the networks in the manner of a semantic network, and use propagation filters to perform dynamic inferencing. PDS Networks are implemented in DCAIN, a natural language understanding system that: (1) uses a simple procedure for determining link weight values, (2) exploits automatic generalization, (3) represents structure, (4) provides knowledge-level parallelism, and (5) drastically reduces training time. Thus, DCAIN has a number of advantages over previous connectionist systems, and integrates the best features of symbolic and PDP approaches while avoiding their associated problems.

References

[Hinton, 1981] G. E. Hinton. Implementing Semantic Networks in Parallel Hardware. In *Parallel Models of Associative Memory*, Lawrence Erlbaum, Hillsdale, NJ, 1981.

[Pollack, 1988] J. Pollack. Recursive Auto-Associative Memory. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, Montreal, 1988.

[Rumelhart and McClelland, 1986] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing*, Volume 1. MIT Press, Cambridge, Massachusetts, 1986.

[Sejnowski, 1981] T. J. Sejnowski. Skeleton Filters in the Brain. In *Parallel Models of Associative Memory*, Lawrence Erlbaum, Hillsdale, NJ, 1981.

[Sumida *et al.*, 1988] R. A. Sumida, M. G. Dyer, and M. Flowers. Integrating Marker Passing and Connectionism for Handling Conceptual and Structural Ambiguities. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, Montreal, 1988.

[Sumida and Dyer, 1989] R. A. Sumida and M. G. Dyer. Storing and Generalizing Multiple Instances while Maintaining Knowledge-Level Parallelism. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, 1989.

[Waltz and Pollack, 1985] D. Waltz and J. Pollack. Massively Parallel Parsing. *Cognitive Science*, 9:51–74, 1985.