Final - As Sent

Unifying Several Natural Language Systems in a Connectionist Deterministic Parser

Stan C. Kwasny, Kanaan A. Faisal, and William E. Ball

Center for Intelligent Computer Systems†
Department of Computer Science
Washington University
St. Louis, Missouri 63130

ABSTRACT

Natural Language systems based on deterministic parsing have demonstrated that Natural Language Processing can be performed deterministically. Although Marcus was first to propose this approach in his MIT doctoral work, others have independently extended this notion to the task of parsing ungrammatical sentences, resolving lexical ambiguities, and acquiring syntactic rules.

We have found it beneficial to combine these tasks into one implementation which is partly symbolic and partly sub-symbolic. The introduction of connectionism has proven crucial to the success of the endeavor. We have constructed a connectionist deterministic parser which possesses capabilities comparable to each of these systems. It combines the concepts and ideas from deterministic parsing with the generalization and robustness of connectionist, adaptive (neural) networks. A back propagation neural network simulator is used in this work. The ultimate goal is to produce a parser that is capable of acquiring some reasonable facility with language without refusing those inputs that are slightly different syntactically from the inputs it is designed to process.

INTRODUCTION

The determinism hypothesis imposes important restrictions on Natural Language Processing (NLP). It proposes that such processing need not depend in any fundamental way on backtracking. PARSIFAL (Marcus 1980) was the first of a number of systems to demonstrate the notion of deterministic parsing of Natural Language. These ideas have been independently extended to the parsing of ungrammatical sentences in PARAGRAM (Charniak 1983), to the resolution of lexical ambiguities in ROBIE (Milne 1986), and to the learning of syntactic rules in LPARSIFAL (Berwick 1985).

As Figure 1 illustrates, these systems share their common roots in deterministic parsing, but represent independent solutions in specific problem areas. The integration of their processing capabilities is a specific goal of our work (Kwasny 1988). The system architecture of PARSIFAL has been re-configured and the behavior of the rules and other mechanisms from these systems are being simulated using a neural network simulator. Learning in the network is achieved through backward propagation (Werbos 1974; Rumelhart et al. 1986) and a prototype implementation of our Connectionist Deterministic Parser (CDP) has been constructed (Kwasny and Faisal 1989).

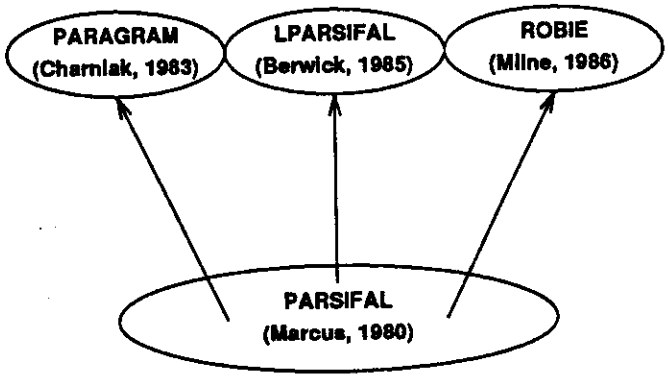

Figure 1:
Deterministic Parsing Systems

In our prototype, a moderate-sized grammar developed from PARSIFAL is used for training. Experiments are conducted to determine the effectiveness of training and to investigate whether the connectionist network generalizes properly to ungrammatical and lexically ambiguous cases. In comparisons with the other deterministic parsing systems, CDP performs favorably. Much of the performance depends on the extent and nature of the training, of course, but our results show that through proper training a connectionist network can indeed exhibit the same behavioral effect as the rules. Furthermore, once trained, the network is efficient, both in terms of representation and execution.

† The sponsors of the Center are McDonnell Douglas Corporation and Southwestern Bell Telephone Company.

ARCHITECTURE OF CDP

CDP is composed of a connectionist network trained using backward propagation from rule templates which are derived from a deterministic grammar. Rule templates are intermediate between symbolic rules and the training patterns required by the network. Each rule template typically represents a large number of patterns. They serve to relate situations that occur during parsing with the action deemed appropriate for that situation. Actions in CDP are performed symbolically on traditional data structures which are also maintained symbolically.

Deterministic ("Wait-and-See") Parsing

Deterministic ("Wait-and-See") Parsers process input sentences primarily left-to-right. Determinism is accomplished by permitting a lookahead of up to three constituents with a constituent buffer designated for that purpose. To permit embedded structures, a stack is also part of the architecture. This is illustrated in Figure 2. The absence of backtracking is an important advantage in developing a connectionist-based parser since structures, once built, need never to be thrown away.

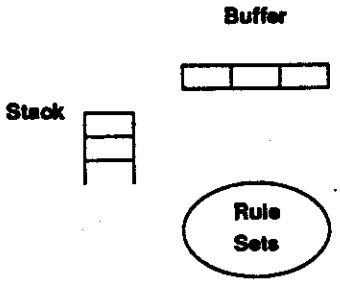

Figure 2:
Wait-and-See Parsing

Rules are partitioned into rule sets and a single processing step consists of selecting a rule that can fire from an active rule set. Rule sets are usually associated with the current (top-level) node of the structure being built. Conflicts are resolved from the static ordering (priority) of rules within the rule set. Once selected, the rule is fired and its action is performed. The action effects changes to the stack and buffer. After a series of processing steps, a termination rule fires and processing is terminated. The final structure is left on top of the stack.

The grammar used in CDP is capable of processing a variety of sentence forms which end with a final punctuation mark. Simple declarative sentences, yes-no questions, imperative sentences, and passives are permitted by the grammar. The model actually receives as input a canonical representations of each word in the sentence in a form that could be produced by a simple lexicon. Such a lexicon is not part of the model in its present form.

System Organization

As Figure 3 illustrates, CDP is organized into a symbolic component and a sub-symbolic component. The latter component is implemented as a numeric simulation of an adaptive neural network. The symbolic and numeric components cooperate in a tightly coupled manner since there are proven advantages to this type of organization (Kitzmiller and Kowalik 1987). For CDP, the advantages are performance and robustness.

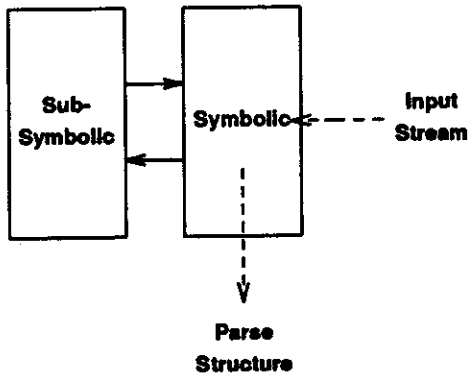

Figure 3:
CDP System Organization

The symbolic component manages the input sentence and the flow of constituents into the lookahead buffer, coding them as required for the input level of the network in the sub-symbolic component. On the return side, it evaluates the activations of the output units, decides which action to perform, and performs that action, potentially modifying the stack and buffer in the process.

The responsibility of the sub-symbolic component, therefore, is to examine the contents of the buffer and stack and yield a preference for a specific action. These preferences are garnered from many iterations of back-propagation learning with instances of the rule templates. Learning itself occurs off-line and is a time-consuming process, but once learned the performance of the system is excellent. Computations flow only in one direction in the network. The feed-forward multiplication of weights and computation of activation levels for individual units produce the pattern of activation on the output level. Activation of output units is interpreted in a winner-take-all manner, with the highest activated unit determining the action to be taken.

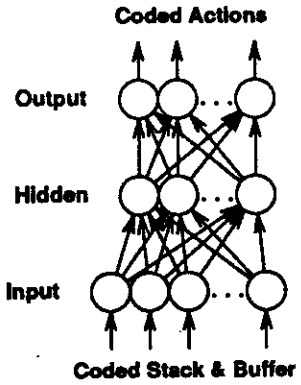

Figure 4:
Sub-Symbolic Component

In the prototype, the network has a three-layer architecture as illustrated in Figure 4, with 37 input units, 20 hidden units, and 20 output units. The input layer consists of four pools of input units. The first three pools represent the buffer, with each containing the features of a buffer item, and the fourth pool represents the top of the stack including the current node of the parse tree. One hidden layer is used in all of our experiments. The output layer represents the 20 actions that can be performed on each iteration of processing.

GRAMMAR LEARNING IN CDP

There are two distinct approaches to training a network to parse sentences. Each of these training strategies result in a slightly different version of CDP. The difference lies in the nature of the training patterns presented. One approach uses rule templates, training patterns derived from the rules. This type of learning is deductive in the sense that a very general form of each rule is learned from which the parser must derive actions specific to individual cases. The second approach uses training data derived from sentence processing traces. This form of training is inductive in the sense that the parser must arrive at general patterns of performance from the specific instances presented.

Deductive Learning

Rule templates are the basis for deductive training. Pattern instances of the rule templates are presented repeatedly and the weights in the network are adjusted at each step. This continues until the network converges and outputs are produced for each rule which are below a threshold of error tolerance. When this occurs, the parser that uses the network should correctly parse exactly those sentences that the original rules can parse. In this way, the rules of PARSIFAL can be simulated in a deductively-trained CDP.

In the canonical input format of a rule template, word forms are represented as a list of syntactic features. The set of possible features is chosen as necessitated by the grammar being coded. In general, each word form is represented by a feature vector in which one or more features are present (indicated by +1 in the feature vector). All other features are either OFF(-1) or DO NOT CARE (?). A rule template is instantiated to form a training pattern by randomly changing ? to either +1 or -1. It is then applied to the input level of the network.

Grammar rules are coded into rule templates by concatenating the feature vectors of the component constituents from the stack and buffer. Each grammar rule takes the following form:

{ <Stack> <1st Item> <2nd Item> <3rd Item> →
Action }

For example, a rule for Yes/No questions would be written:

{ < *S node* > < *"have"* > < *NP* > < *VERB, -en* > →
Switch 1st and 2nd items }

while a rule for imperative sentences would be written:

{ < *S node* > < *"have"* > < *NP* > < *VERB, inf* > →
Insert YOU }

By replacing each constituent with its coding, a rule template is created. Each rule in the grammar gives rise to one or more rule templates. In the two rules above, rule templates are created with a ? value for many of the specific verb features of the initial form "have" in each rule, but are carefully coded for the differences in the third buffer position where the primary differences lie. Because different actions are required, these are also coded to have different teaching values during training. Organization of the templates into packets or some other form of grouping is not necessary here as in the deterministic grammar.

The probability of a ? becoming a +1 or -1 is equal and set at 0.5. All weights in the network are initialized to random values between -0.3 and +0.3. After the presentation of each pattern, an error signal is derived by comparing activation on the output layer (the network's prediction) with the desired output pattern (the rule template's action). That error signal is back-propagated through all the connections and the weights adjusted before presenting the next pattern.

A sentence is parsed by iteratively presenting the network with coded inputs and performing the action specified by the network. Each sentence receives a score representing the overall average strength of responses during processing. The score for each processing step is computed as

the reciprocal of the error for that step. The error is computed as the Euclidean distance between the actual output and an idealized output consisting of a -1 value for every output unit except the winning unit which has a +1 value. The errors for each step are summed and averaged over the number of steps. The average strength is the reciprocal of the average error per step.

Inductive Learning

A second type of training for the network uses training patterns derived from traces of the situations encountered and actions performed during the processing of actual sentences. This processing is guided by application of the rules of a deterministic grammar as before. PARSIFAL is simulated in this way and the task of the inductively-trained CDP parallels that of LPARSIFAL.

In LPARSIFAL the object is to learn (symbolic) grammar rules from examples of correct sentences. The success of this task is gauged by directly comparing the rules learned to those of PARSIFAL. LPARSIFAL succeeds in learning approximately 70% of those rules from a carefully constructed set of training sentences.

In CDP inductive training requires the network to exhibit the correct rule-following behavior after being trained with a sample of sentence traces. Training occurs through the mechanism of backward propagation which is a general purpose network training algorithm. No symbolic rules are learned as such, but the behavioral characteristics of the rules are captured within the parameters of the network. Furthermore, the behavior is guaranteed to approximate the behavior required in the sample training sentences as closely as desired, depending on the convergence rate and the quantity of training employed.

CDP, therefore, can exhibit different properties depending on the patterns used in training. Inductive learning is a much more tedious process since much more data is required as compared to that required for deductive training. Also, the range of sentence types handled depends greatly on the completeness of the examples presented. Deductive training imposes an ordering on the training patterns that assures a completeness which is difficult to achieve with inductive training, but inductive training patterns reflect the frequency of rule occurrences seen in actual sentence processing.

SIMULATION OF NLP SYSTEMS

Each of the three NLP systems mentioned provide improvements and enhancements to the PARSIFAL system. Since these improvements are complementary, it is desirable to combine their features into one system. Although we have chosen to pursue this goal through connectionism, could this be accomplished symbolically?

Attempting to combine these systems symbolically would require that several issues be addressed. PARAGRAM handles ungrammatical sentences by changing some of the rules and introducing a scoring mechanism. Since ROBIE adds rules for dealing with lexical ambiguity, these rules would have to be examined and made compatible with the scoring facility of PARAGRAM. Furthermore, rule-learning in LPARSIFAL would have to be extended to give it the capability of learning these new rules even though, at present, it is only capable of learning a percentage of the rules in PARSIFAL. Thus, for the symbolic approach to yield one integrated system the combinatorial interplay among the parts must be addressed. While not impossible, the building of such a system would certainly involve solving some difficult problems.

The connectionist approach has several important advantages in unifying these four systems. Much of the capability is derived directly from properties of connectionist networks and therefore no special mechanisms are required for each. Learning is a fundamental feature of the approach. Ungrammatical sentences, comparable to those of PARAGRAM, are processed as a consequence of the generalization properties of the network. These same properties also aid in the disambiguation of lexical items.

Grammatical Sentences

Experimentation with grammatical sentences demonstrates the ability of CDP to perform as PARSIFAL. Thus, the claim that CDP simulates both PARSIFAL and LPARSIFAL is substantiated. For example, the following grammatical sentences were processed by the system just as the rule-based grammar would process them.

(1)	John should have scheduled the meeting.	283.3
(2)	Has John scheduled the meeting?	132.2
(3)	The boy did hit Jack.	298.2
(4)	Schedule the meeting.	236.2
(5)	Tom hit(v) Mary.	485.0
(6)	Tom will(aux) hit(v) Mary.	547.5
(7)	They can(v) fish(np).	485.0
(8)	They can(aux) fish(v).	598.2

The numbers indicate the average strength computed for each sentence, but should be used for comparison purposes only. Each example shows a high average strength value, indicating that the rules used in training have been learned.

Ungrammatical Sentences

PARAGRAM extends PARSIFAL to handle ungrammatical sentences. This is accomplished by considering all rules in parallel and scoring each test performed on the left-hand side of a rule according to predefined weights. The rule with the best score fires. In this way, processing will always have some rule to fire. Reported experimentation with PARAGRAM shows this to be an effective method of stretching the inherent capabilities of the grammar.

For CDP, an important test of its generalization capabilities is its response to novel sentences. This is strictly dependent upon its experience since no relaxation rules (Kwasny and Sondheimer 1981), meta-rules (Weischedel and Sondheimer 1983), or other special mechanisms were added to the original grammar to handle such cases. This experiment consists of testing a few ungrammatical sentences that are close to the training data and within the scope of our encoding. For example the following ungrammatical sentences result in reasonable structures.

(9)	*John have should scheduled the meeting.	25.1
(10)	*Has John schedule the meeting?	38.1
(11)	*John is schedule the meeting.	4.7
(12)	*The boy did hitting Jack.	26.6

Sentence (9), for example, results in the structure shown in Figure 5. The numbers again indicate the average strength for each sentence. In our view, this is a reasonable structure for this sentence. The only observable effect is that overall average strength is lower for ungrammatical sentences when compared to similar grammatical ones.

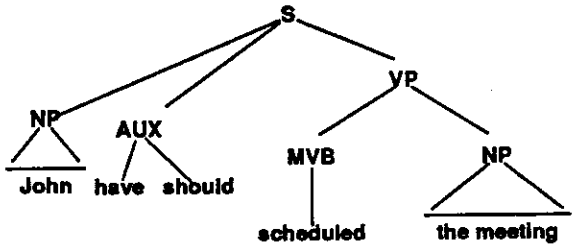

Figure 5:
Parse of "*John have should scheduled the meeting."

Selected ungrammatical sentences are properly processed as a direct result of the generalization properties of learning. When presented an ungrammatical sentence, that is one for which the rules of the grammar would fail to find a parse, the network automatically relates the situations arising during parsing to similar situations on which it has been trained. The tendency is for it to select the closest situation. Very often, this generates precisely the response required to relate the deviant form to one that is not.

Lexically Ambiguous Sentences

ROBIE extends PARSIFAL to address issues of lexical ambiguity. Many of these require additional rules and lexical features in ROBIE to properly handle. In the deterministic approach, it is important that lexical items be properly disambiguated to permit processing to proceed without backtracking.

Thus, in another set of experiments with CDP, the parser was tested for this. Normal sentences were presented, except that selected words were coded ambiguously (here indicated by angle brackets < > around the word) to represent an ambiguously stored word from the lexicon. Consider the following sentences:

(13)	Tom <will> hit Mary.	118.7
(14)	Tom <hit> Mary.	39.0
(15)	They <can> fish(np).	4.5
(16)	They can(aux) <fish>.	172.2

The numbers again indicate the average strength for each sentence. In the cases shown, the lexically ambiguous words were correctly interpreted and reasonable structures resulted. For example, sentence (15) resulted in the same structure as sentence (7) produced. This is shown in Figure 6. Likewise, sentence (16) produced the same structure as sentence (8). This is the structure in Figure 7. We note again that the overall average strengths were lower than comparable grammatical sentences discussed, as expected.

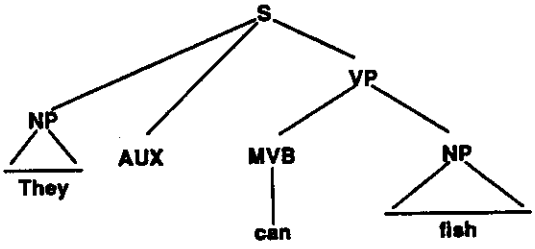

Figure 6:
Parse of "They can(v) fish(np)"

CDP utilizes syntactic context to resolve similar ambiguities. Again, the generalization capability of the network automatically works to relate novel situations to its training cases. For lexically ambiguous situations, some inputs may contain features which confuse its identity as expected by the parser. The context provided by the buffer and stack of the deterministic parser has proven to be

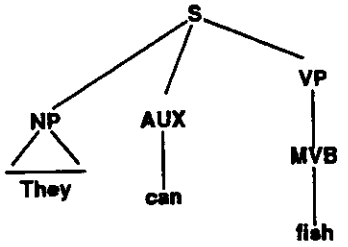

Figure 7:
Parse of "They can(aux) fish(v)"

sufficient to aid in resolving most ambiguities. An important fact is that no additional rules or mechanism is required to provide this capability.

DISCUSSION

In its present stage of development, CDP clearly encompasses many of the capabilities of PARSIFAL, LPARSIFAL, PARAGRAM, and ROBIE. The extent to which it measures up to each is under investigation (Faisal 1989), but in our prototype, the capability for each line of research to continue in a unified fashion has been demonstrated and verified. We have argued that it would be very difficult to combine these systems symbolically.

The advantages of this effort are clear. The resulting system possesses many more types of language coverage than any of the four systems individually. To achieve this symbolically would require the extension of the mechanisms in each of the systems to include the other. It is difficult to see how this would be done. Many of the processing capabilities of CDP are derived from the introduction of connectionism and from the learning technique employed. CDP critically depends on the network, how it is trained, and the sophistication of the grammar on which training is based. No special mechanisms are required for ungrammaticality and lexical ambiguity. If the network generalizes well, these capabilities follow naturally.

In the prototype, the system has been limited, in a sense, by only training with a moderate-sized grammar. In doing so, the capabilities that CDP possesses have been established. As CDP is trained using more examples from a larger grammar, its capabilities are expected to improve and equal or surpass those of the four systems. This is our goal in current and future work.

ACKNOWLEDGMENTS

We are grateful to many people for helping in this effort. The authors express gratitude to Georg Dorffner, David Harker, Dan Kimura, Ron Loui, and John Merrill for thoughtful discussions and comments concerning this work.

REFERENCES

Berwick, R.C. 1985. *The Acquisition of Syntactic Knowledge*. MIT Press, Cambridge, MA.

Charniak, E. 1983. "A Parser with Something for Everyone." In *Parsing Natural Language*, M. King, ed. Academic Press, New York, NY, 117-150.

Faisal, K.A. 1989. "CDP: Connectionist Deterministic Parser, A Dissertation Proposal" Technical Report WUCS-89-33. Department of Computer Science, Washington University, St. Louis, Mo. (Aug.).

Kitzmiller, C.T., and J.S. Kowalik. 1987. "Coupling Symbolic and Numeric Computing in Knowledge-Based Systems." *AI Magazine 8*, no. 2, 85-90.

Kwasny, S.C. and K.A. Faisal. 1989. "Competition and Learning in a Connectionist Deterministic Parser." In *Proceedings of the 11th Annual Conference of The Cognitive Science Society*, (Ann Arbor, MI, Aug. 16-19). Lawrence Erlbaum Associates, Hillsdale, NJ, 690-697.

Kwasny, S.C. and N.K. Sondheimer. 1981. "Relaxation Techniques for Parsing Ill-Formed Input." *American Journal of Computational Linguistics 7*, no. 2, 99-108.

Kwasny, S.C. 1988. "A Parallel Distributed Approach to Parsing Natural Language Deterministically." Technical Report WUCS-88-21. Department of Computer Science, Washington University, St. Louis, Mo. (Aug.).

Marcus, M. P. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA.

Milne, R. 1986. "Resolving Lexical Ambiguity in a Deterministic Parser." *Computational Linguistics 12*, no. 1 (Jan-Mar): 1-12.

Rumelhart, D. E., G. Hinton, and R.J. Williams. 1986. "Learning Internal Representations by Error Propagation." In *Parallel Distributed Processing*, D.E. Rumelhart and J.L. McClelland, MIT Press, Cambridge, MA, 318-364.

Weischedel, R.M. and N.K. Sondheimer. 1983. "Meta-Rules as a Basis for Processing Ill-Formed Input." *American Journal of Computational Linguistics 9*, no. 3-4, 161-177.

Werbos, P. 1974. "Beyond Regression: New Tools for Prediction and Analysis in Behavioral Science." PhD Thesis. Harvard University, Cambridge, Ma.