Evolution of a Hybrid Deterministic Parser

Kanaan A. Faisal and Stan C. Kwasny

Center for Intelligent Computer Systems[†]

Department of Computer Science

Washington University

St. Louis, Missouri 63130

1. Introduction

A hybrid deterministic parser is under development. It combines the notions of deterministic parsing initiated by Marcus (1980) in the PARSIFAL system with that of connectionism. The result is a parser which is decidedly more robust than PARSIFAL (Marcus, 1980) and which compares favorably with various of its extensions. Our Connectionist Deterministic Parser (CDP) represents a departure from traditional deterministic parsers in its combination of both symbolic and sub-symbolic (connectionist) components. The symbolic component manages the stack as well as the flow of sentence elements into the buffer while the sub-symbolic component decides how these structures should be managed and receives its training based on patterns derived from the rules of a deterministic grammar.

Three generations of experiments are described. In each set of experiments the size of the grammar increases as does the complexity of the task. In scaling up, similar results are found for each of the grammars. Our training techniques, therefore, are shown to be applicable to successively larger subsets of English. The approach permits some simplifications over traditional deterministic parsers, including the elimination of rule packets and priorities. Furthermore, parsing is performed more robustly by the sub-symbolic component and with more tolerance for error in the parsing process. Data are presented which show how a neural network trained with linguistic rules can parse grammatical, ungrammatical, and lexically ambiguous sentences.

2. Determinism and NLP

If we accept the determinism hypothesis posed by Marcus (1980) in his work on PARSIFAL, it must follow that Natural Language Processing (NLP) need not depend in any fundamental way on backtracking. As a further consequence, no partial structures need be produced during parsing which fail to become part of the final structure. Extensions to PARSIFAL have been researched independently including the parsing of ungrammatical sentences in PARAGRAM (Charniak, 1983), the resolution of lexical ambiguities in ROBIE (Milne, 1986), and the acquiring of syntactic rules from examples in LPARSIFAL (Berwick, 1985). The three extensions to PARSIFAL are all derivatives of deterministic parsing, but represent independent solutions in specific problem areas. The integration of their processing capabilities is one goal of our work (Kwasny et al., 1990). The ultimate goal is to produce a parser that is capable of learning some reasonable facility with language, but does not fail on inputs that are only slightly different from the inputs it is designed to process.

Determinism in PARSIFAL is accomplished by permitting lookahead of up to three constituents within a buffer designated for that purpose. A stack permits embedded structures and facilitates processing. Rules are partitioned into packets which become active or inactive during

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

parsing, but are usually associated with the current (top-level) node of the structure being built. A single processing step consists of selecting a rule from an active rule packet and firing the rule. Conflicts are resolved from the static ordering (priority) of rules within the packet. 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.

3. Connectionist Deterministic Parsing

As Figure 1 illustrates, CDP is organized into a symbolic component and a sub-symbolic component. The latter component is simulated by an adaptive neural network. The two 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 competition and robustness (see Kwasny and Faisal, 1989).

The sub-symbolic component of CDP contains a connectionist network trained using backward propagation (Werbos 1974; Rumelhart et al, 1986) from rule templates which are derived from the 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. Actions in CDP are performed symbolically on traditional data structures which are also maintained symbolically.

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 examines the activations of the output units, decides which action to perform, and performs that action. It is the responsibility of the sub-symbolic component, therefore, to 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

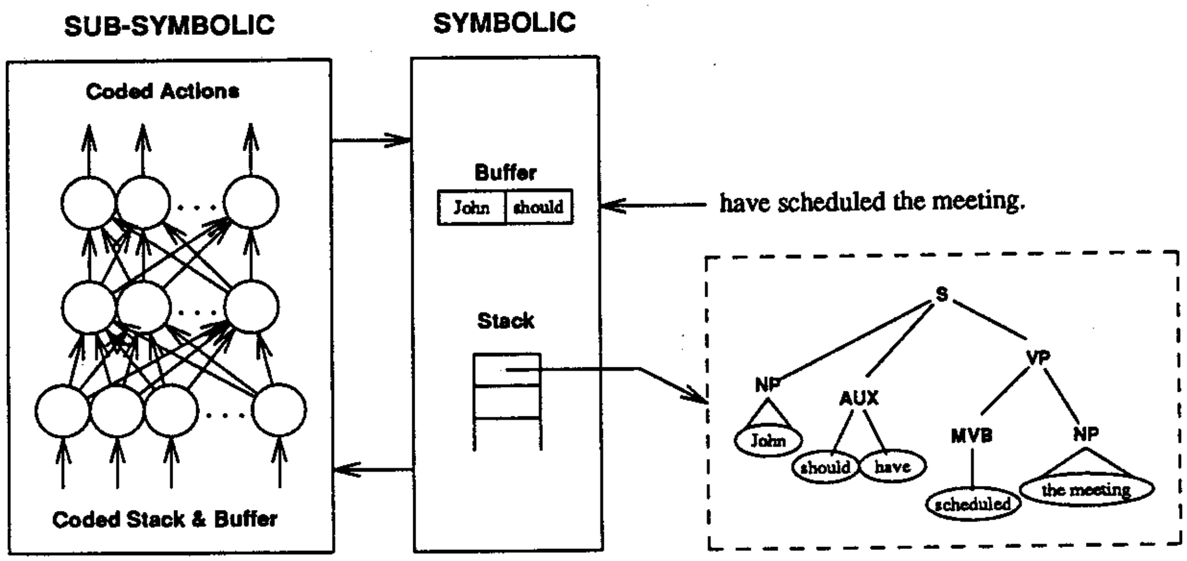

Figure 1: CDP System Overview

occurs off-line and is time-consuming, but once learned the system is efficient. 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.

During sentence processing, the network is presented with encodings of the buffer and the top of the stack. The model does not actual see the raw sentence but a canonical representation of each word in a form that could be produced by a simple lexicon, although such a lexicon is not part of the model in its present form. The network produces the action to be taken which is then performed. If the action creates a vacancy in the buffer and if more of the sentence is left to be processed then the next sentence component is moved into the buffer. The process then repeats until a stop action is performed, usually when the buffer becomes empty. Iteration over the input stream is achieved in this fashion.

When a sentence form like ''John should have scheduled the meeting'' appears in the input stream, the first two constituents fill the buffer as shown in Figure 1. These contents along with the contents of the top of the stack and its attachments are encoded and presented to the network. The network, in turn, produces a single action which is then executed symbolically. This process repeats until a stop action is performed at which time the resultant parse structure is left on top of the stack.

Training of CDP proceeds by presenting patterns to the network and teaching it to respond with an appropriate action. The input patterns represent encodings of the buffer positions and the top of the stack from the deterministic parser. The output of the network contains a series of units representing actions to be performed during processing and judged in a winner-take-all fashion. Network convergence is observed once the network can achieve a perfect score on the training patterns themselves and the error measure has decreased to an acceptable level (set as a parameter).

Our neural network simulator features a logistic function that computes values in the range of −1 to +1. Each grammar rule is coded as a training template which is a list of feature values. In general, each constituent is represented by an ordered feature vector in which one or more values is ON(+1) for features of the form and all other values are either OFF(−1) or DO NOT CARE (*?*). A rule template is instantiated by randomly changing *?* to +1 or −1. Thus, each template can be instantiated to give many training patterns and each training epoch is slightly different. It is obviously impossible to test the performance of all these cases, so for the purpose of judging convergence, a zero is substituted for each *?* in the rule template to provide testing patterns. For more discussion of the training process, see Faisal and Kwasny (1990).

4. Experimentation

Our experimentation has examined three very different target grammars, which we shall call small, medium, and large. Figure 2 shows some of the characteristics of these grammars for comparison. The small grammar is based on an example from the Winston AI text (1984). The network requires 44 input units to encode the stack and three-place buffer. Our choice of 15 hidden units is determined empirically. The medium and large grammars, based loosely on appendices C and D in Marcus (1980), contain 22 and 73 rules respectively. The network configurations reflect an increase from 5 to 20 to 40 actions. A variety of training runs have been made with each grammar. Shown is the number of presentations of training patterns sufficient to get good convergence and generalization results, although fewer presentations often give satisfactory results also. The

	Target Grammars		
	Small	Medium	Large
Number of Rules	13	22	73
Number of Actions	5	20	40
Network Size	44-15-5	35-20-20	66-40-40
Presentations (× 1000)	200	500	1000
Based On	Example (Winston, 1984)	Appendix C (Marcus, 1980)	Appendix D (Marcus, 1980)

Figure 2: Summary of Target Grammars Used in CDP

large grammar is the major focus of a forthcoming Ph.D. thesis as proposed by Faisal (1989).

In our initial attempt to demonstrate the feasibility of our approach, a small, simple grammar is used. It features S, NP, VP, and PP structures with an assumed preprocessing for noun phrases. Coding of three rule packets (S, VP, and PP) as rule templates provides training data for the 44-15-5 unit network. Perfect performance, as determined by presentation of a limited number of test sentences, is achieved for the 13 grammar rules coded. No generalization experiments were performed due to the limited nature of the grammar.

A second set of experiments shows how a variety of more complicated mechanisms essential to PARSIFAL are realized in our architecture. The medium grammar is much more sophisticated than the small one and permits reasonable generalization experiments to be conducted. It is capable of processing a variety of simple sentence forms such as simple declaratives, simple passives, imperative sentences, and yes-no questions. Appendix C of Marcus (1980) serves as the model for the rules of this grammar. It represents all of the basic mechanisms of deterministic parsing as discussed for PARSIFAL.

In a third set of experiments, a much larger and more general grammar is used. In this case, the grammar consists of 73 rules and represents rules for parsing many sentence forms such as simple declarative sentences, passives, imperatives, yes-no questions, wh-questions, wh-clauses, and other embedded sentences.

With the medium and large grammars, several sentences are coded for testing and comparison purposes. Some would parse correctly by the rules of the deterministic parser, while others are mildly ungrammatical and lexically ambiguous. Most of these examples are drawn from work cited earlier by Charniak and Milne. In parsing the sentences, the performance of the network is measured in two ways: first, by the validity of the structure produced; and second, by the average strength of the response of the neural network. Strength is measured as the reciprocal of the average error for each step. In this way, it is determined if the network is generalizing in any useful way and whether its responses are being challenged by other ones. Several dozen sentences have been examined and tested with each grammar and desirable generalization properties have been shown.

In examining grammars of varying sizes, our objective is to determine whether the same generalization properties seen in the medium grammar would scale up to a much larger and more realistic set of grammar rules. Our data supports this conclusion.

5. Summary

With our hybrid architecture for connectionist deterministic parsing, the functionality of symbolic deterministic parsing is demonstrated in addition to other capabilities. Training becomes difficult as the complexity of the grammar increases, but many of the advantages of the connectionist approach can be realized by modifying a deterministic parser as described. Likewise, many of the drawbacks of previous connectionist attempts at NLP can be overcome, while addressing some of the deficiencies of symbolic models. Thus, the marriage is a good one from our perspective.

Finally, since CDP is based on rules, there are general lessons for rule-based systems. In particular, if robustness, generalization, and other properties of connectionism are desired in an expert system, our approach suggests a method for achieving those properties. This is further discussed in Kwasny and Faisal (in press).

References

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

Charniak, E. 1983. A Parser with Something for Everyone. In M. King, (Ed.), *Parsing Natural Language*. New York, NY: Academic Press.

Faisal, K.A. and S.C. Kwasny. 1990. Deductive and Inductive Learning in a Connectionist Deterministic Parser. In *Proceedings of the International Joint Conference on Neural Networks* (Vol. 2, pp.471-474). Hillsdale, NJ: Lawrence Erlbaum Associates.

Faisal, K.A. 1989. *CDP: Connectionist Deterministic Parser* (Technical Report WUCS-89-33). St. Louis, MO: Washington University, Department of Computer Science.

Kitzmiller, C.T., and J.S. Kowalik. 1987. Coupling Symbolic and Numeric Computing in Knowledge-Based Systems. *AI Magazine, 8*, 85-90.

Kwasny, S.C., K.A. Faisal, and W.E. Ball. 1990. Unifying Several Natural Language Systems in a Connectionist Deterministic Parser. In *Proceedings of the Western Multi Conference: Artificial Intelligence and Simulation*, (FORTHCOMING).

Kwasny, S.C., and K.A. Faisal. (in press). Rule-Based Training of Neural Networks. *Expert Systems with Applications* (Special Issue on Applying Artificial Neural Networks to Expert Systems). Elmsford, NY: Pergamon Press.

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*. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Milne, R. 1986. Resolving Lexical Ambiguity in a Deterministic Parser. *Computational Linguistics, 12*, 1-12.

Rumelhart, D. E., G. Hinton, and R.J. Williams. 1986. Learning Internal Representations by Error Propagation. In D.E. Rumelhart and J.L. McClelland & the PDP Research Group (Eds.), *Parallel Distributed Processing* (Vol. 1). Cambridge, MA: MIT Press.

Werbos, P. 1974. *Beyond Regression: New Tools for Prediction and Analysis in Behavioral Science*. Unpublished doctoral dissertation, Harvard University, Cambridge, MA.

Winston, P.H. 1984. *Artificial Intelligence* (2nd ed.). Reading, MA: Addison-Wesley.