Deterministic Parsing of English: A Case for Sub-Symbolic Learning

Kanaan A. Faisal

Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran 31261
Kingdom of Saudi Arabia
(03) 860-2175
kaf8821@cics.wustl.edu

Stan C. Kwasny

Center for Intelligent Computer Systems[†]
Department of Computer Science
Washington University
Campus Box 1045, Bryan 509
One Brookings Drive
St. Louis, MO 63130-4899
U.S.A.
(314) 889-6123
sck@wucs1.wustl.edu

Summary

A Connectionist Deterministic Parser (CDP) extends previous symbolic work by introducing a sub-symbolic component to replace the English parsing rules. Learning is achieved in the neural network through backward error propagation. A more robust parser is the result — one which is capable of processing a wider variety of sentence forms. Data are presented which demonstrate its capabilities for parsing grammatical as well as ungrammatical and lexically ambiguous sentences.

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

Deterministic Parsing of English:

A Case for Sub-Symbolic Learning

Kanaan A. Faisal

Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran 31261
Kingdom of Saudi Arabia
(03) 860-2175
kaf8821@cics.wustl.edu

Stan C. Kwasny

Center for Intelligent Computer Systems[†]
Department of Computer Science
Washington University
Campus Box 1045, Bryan 509
One Brookings Drive
St. Louis, MO 63130-4899
U.S.A.
(314) 889-6123
sck@wucs1.wustl.edu

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

1. Introduction

A Connectionistic Deterministic Parser (CDP) extends previous symbolic work by introducing a sub-symbolic component that replaces the English parsing rules. Learning is achieved in the neural network through backward error propagation. A more robust parser results — one which is capable of processing a wider variety of sentence forms. Data are presented which demonstrate its capabilities for parsing grammatical as well as ungrammatical and lexically ambiguous sentences.

The determinism hypothesis which forms the basis for PARSIFAL (Marcus, 1980) imposes important restrictions on Natural Language Processing (NLP). NLP need not depend in any fundamental way on backtracking. Garden path sentences, of course, are an exception to this. Extensions to PARSIFAL have been researched independently including the acquiring of syntactic rules from examples in LPARSIFAL (Berwick, 1985), the parsing of ungrammatical sentences in PARAGRAM (Charniak, 1983), and the resolution of lexical ambiguities in ROBIE (Milne, 1986). The integration of the processing capabilities of these systems is one goal of our work (Kwasny et al., 1990).

2. Connectionist Deterministic Parsing

CDP combines the concepts and ideas from deterministic parsing together with the generalization and robustness of connectionist, adaptive (neural) networks. 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. Our experiments have examined two different approaches to grammar learning and compared them (Faisal and Kwasny, 1990).

Parsing experiments are conducted to determine the effectiveness of training and to investigate whether the connectionist network generalizes properly in ungrammatical and lexically ambiguous cases. In comparison with 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.

Some modification to the deterministic grammar rules seen in PARSIFAL and LPARSIFAL are necessary to insure the suitability of each rule for use with our "winner-take-all" network. These changes are minor and in some cases greatly simplify the rules. They are made without altering the capabilities represented in the original set of grammar rules. Some of these changes have been proposed by others. The basic changes to the rules are: elimination of the packet system; removal of the attention-shifting mechanism; removal of rule priorities; reduction of lookahead to two positions instead of three; and revision of the rules so that a single action is performed by each. In principle, only the last change is required.

Actions such as creating and attaching or selecting the argument structure of the verb are carried out symbolically in CDP. Also, a symbolic lexicon is consulted to determine the properties of words. When a predicate such as a verb is encountered, the requirements or expectations for its arguments are made part of the features of the active VP node, thus affecting which actions will be executed later on. It should be noted that this simplification strategy parallels that of modern linguistic theory in that specific conditions on rule application have been "factored out" into general constraints that need to be learned.

CDP is capable of successfully processing a wide variety of sentence forms such as simple declarative sentences, passives, imperatives, yes-no questions, wh-questions, wh-clauses, and other embedded sentences. The grammar to be learned by the subsymbolic system, which has 73 rules, can be separated into base phrase structure rules and transformational-type rules. The base structure system can be further broken down into rules for NPs, VPs, auxiliaries, main sentence, PPs, and embedded sentences. Transformational rules fall into two groups: simple local transformations (like subject-aux inversion) and major movement rules like *wh* movement. Rules for analyzing verb phrases discriminate among verbs that take different kinds of complements. For example, verbs that take a *wh* complement are discriminated from ones that take a *that* complement.

3. Architecture of CDP

CDP is composed of a connectionist network trained using backward propagation (Werbos 1974; Rumelhart et al, 1986) from rule templates derived from the deterministic grammar. Rule templates are intermediate between symbolic rules and the training patterns required by the network in that a rule

template encodes a rule, but typically represents a large number of training 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.

As Figure 1 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.

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 processing times for the system are excellent. Computations need only flow 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.

In the set of experiments described here, the network has a three-layer architecture, as illustrated, with 66 input units, 40 hidden units, and 40 output units. Each input pattern consists of two feature vectors from the buffer items and one vector from the stack. The first vector activates 26 input units and the second vector activates 12 input units in a pattern vector representing a word or constituent of the sentence. The stack vector activates 28 units representing the current node on the stack and its immediate attachments. One hidden layer has proven sufficient in all of these experiments. The output layer permits the choice of one out of 40 possible actions that can be performed on a single iteration of processing. Further details of the architecture are available in (Kwasny and Faisal, 1989a).

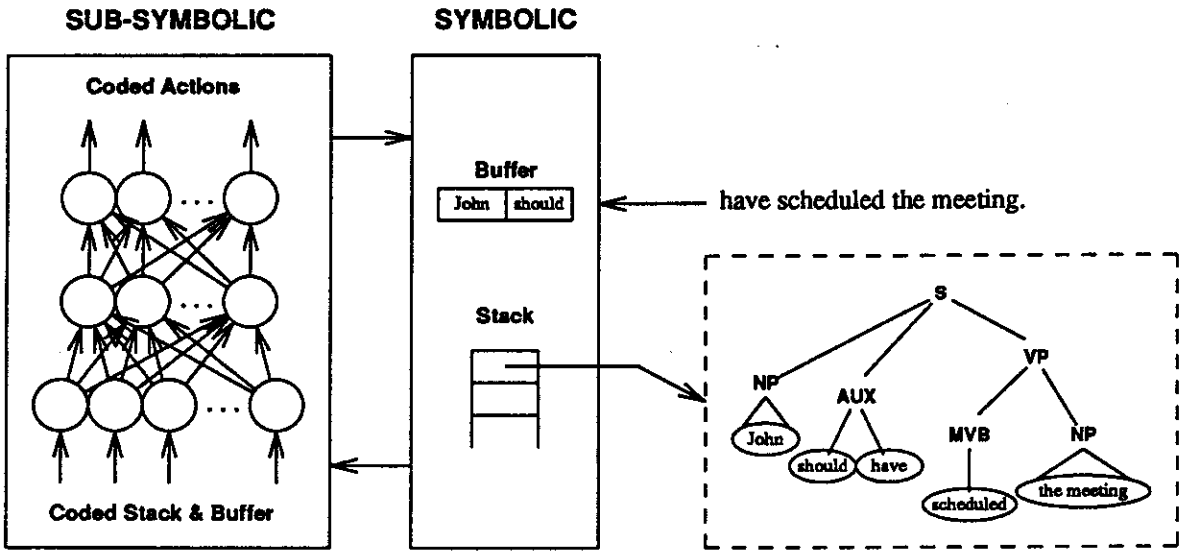

Figure 1: CDP System Overview

During sentence processing, input patterns represent encodings of the buffer items and the top of the stack. What the network actually sees in the input layer is not a sequence of words but a canonical representation of each word in the sentence in a form that could be produced by a simple lexicon, although such a lexicon is not part of the implementation presently. The output layer of the network contains a series of units representing possible actions to be performed during processing and judged in a winner-take-all fashion. If an 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.

From the figure, the nature of the processing can be seen. When a sentence form like "John should have scheduled the meeting." appears in the input stream, the first two constituents fill the buffer. 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, yielding changes in the buffer and stack. This process repeats until a stop action is performed at which time the resultant parse structure is left on top of the stack.

4. Learning a Rule-Based Grammar

Training of CDP proceeds by presenting patterns to the network and teaching it to respond with an appropriate action. Network convergence is observed once the network can achieve a perfect performance on the training patterns themselves and the error measure has decreased to an acceptable level. 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. For more discussion of the learning process, see Faisal and Kwasny (1990).

Berwick (1979; 1985) describes LPARSIFAL, a grammar acquisition system that represents grammatical knowledge as a set of rules. LPARSIFAL begins with a knowledge of X-bar theory (Chomsky, 1980) and an interpreter for applying grammar rules to parse sentences. LPARSIFAL attempts to parse a new sentence using rules it already knows. If it reaches a point at which blockage occurs and no rule can apply, the system attempts to create a new rule that will handle the situation. In composing the rule, conditions are derived from the state of the parse at the point of blockage. The top of the stack and the contents of the input buffer determine those conditions. Upon adding the new rule to memory, existing rules are checked for identical actions. When such rules are found, they may be generalized into a new rule if the conditions are similar.

CDP, unlike LPARSIFAL, does not learn rules but it requires the network to exhibit the correct rule-following behavior after being trained. Training occurs through the mechanism of backward propagation. 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, as closely as desired, the behavior suggested by the rule template (in the case of deductive training) or that required in the sample training sentences (in the case of inductive training). Success depends partially on the size of the network (e.g, number of hidden units), partially on the convergence rate, and partially on the quantity of training employed.

There are two distinct approaches to training a network to parse sentences. Each of these training strategies results in a slightly different version of CDP. The difference lies in the nature of the training patterns presented. The "deductive" strategy uses rule "templates" derived from the rules of a deterministic grammar. It is deductive in the sense that it is based on general knowledge in the form of rules although the resultant network must process specific sentences. The "inductive" strategy derives its training sequence from coded examples of sentence processing. It is inductive in the sense that it is based on traces of the processing of specific sentences but must generalize to a wider range of examples. The goal of both deductive and inductive learning is to produce a network capable of mimicking the rules or sentence processing on which its training is based and to do so in a way that generalizes to many additional cases.

In deductive training, each grammar rule is coded as a training template which is a list of feature values, but templates are not grouped into packets. 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 with equal probability of 0.5. Thus, each template represents many training patterns and each

training epoch is slightly different. During training, the network learns the inputs upon which it can rely in constructing its answers and which inputs it can ignore. For more discussion on training, see Kwasny and Faisal (1989b).

Inductive training uses training patterns derived from traces of the states 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. In many ways, 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 and its success is gauged by direct comparison with the rules of PARSIFAL. Similarly, in CDP, traces showing the processing of correct sentences must generate the proper sequence of actions and success is determined by the appropriateness of the resultant structure. Additionally, CDP must show generalization to novel sentences.

Deductive training generally performs well on all generalization tasks and outperforms inductive training by scoring generally higher on all experiments. Reasons for this include the specificity of the inductive training data as well as the lack of a large amount of training data in the inductive case required to provide sufficient variety. 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.

5. Performance

For testing purposes, several sentences are coded that would parse correctly by the rules of the deterministic parser. Additionally, several mildly ungrammatical and lexical ambiguous sentences are coded to determine if the network generalizes in any useful way. Most of these examples are drawn from work cited earlier by Berwick, Charniak, and Milne. The objective is to discover exactly how syntactic context can aid in resolving such problems.

Experimentation with grammatical sentences confirms that indeed the rules from the grammar have been learned sufficiently to parse sentences. When training with the rule templates, testing for convergence is possible by changing each **?** to a zero value. Here the performance of CDP is examined with actual sentences and the results substantiate the claim that CDP simulates both PARSIFAL and LPARSIFAL.

Grammatical sentences, by our definition, are those which parse correctly in the rule-based grammar from which the training set is derived. Table 1 shows several examples of grammatical sentences which are parsed successfully along with their response strengths. These strengths are computed as the reciprocal of the average error per processing step for each sentence and reflect the certainty with which individual actions for building structures are being selected. Although there is no real meaning in the values of these numbers, they are a useful means of comparing responses during sentence processing. Each example shows a relatively high average strength value, indicating that the training data has been learned well. Parse trees are developed which are identical to ones produced by other deterministic parsing systems. Sentences containing words shown followed by parentheses are presented to CDP unambiguously, even though these words have ambiguous uses. The lexical choices are shown in parentheses.

Capabilities described above only duplicate what can be done rather comfortably symbolically. Of course, the feedforward network in CDP allows very fast decision making due to the nature of the model. But what other features does the model possess? Importantly, how robust is the processing?

PARAGRAM symbolically extends deterministic parsing to ungrammatical sentences by a rule-scoring algorithm. To demonstrate its generalization capabilities, CDP is tested with several examples of ungrammatical sentences. Its performance is strictly dependent upon its training experiences since no relaxation rules (Kwasny and Sondheimer, 1981), meta-rules (Weischedel and Sondheimer, 1983), or other special mechanisms were added to the original grammar rules to handle ungrammatical cases. In Table 2, ungrammatical sentences used in testing are shown along with their average strengths. These examples produce reasonable structures when presented to our system. Note that overall average strength is lower for ungrammatical sentences when compared to similar grammatical ones.

In sentence (1b), for example, the structure produced was identical to that produced while parsing sentence (1a). The only difference is that the two auxiliary verbs, *have* and *should*, were reversed in the parse tree. Such scrambling of words is beyond the capabilities of PARAGRAM as pointed out by

Charniak (1983, p.138). Sentence (2b) contains a disagreement between the auxiliary *has* and the main verb *schedule* and but parsed identical to sentence (3a). Sentences (3b) and (4b) parse comparable to sentence (26a). Sentence (4b), in fact, demonstrates how the presence of extra words does not deter CDP as it did PARAGRAM (Charniak, p.138). Another example from PARAGRAM is sentence (5b) which is processed as if it were progressive tense ('The boy is hitting Jack'). When this sentence is presented to PARAGRAM, a nonsensical parse structure is produced for this sentence as reported by Charniak (1983, p. 137). In CDP it produced a structure like that of sentence (27a), but there is not one clear choice for how the sentence should appear if grammatical. The problems with using a syntax-based approach to handling ungrammatical sentences are well-known (see, for example, Kwasny, 1980). Finally, sentence (6b) produced a very strong response, but the comparable grammatical sentence (5a) produces an even stronger response. Sentences like (6b) are commonly misspoken forms of English.

As a further test of the generalization properties of CDP, sentences containing lexically ambiguous words are tested. Some of these sentences are shown in Table 3. Of course, ROBIE takes a symbolic approach in extending PARSIFAL to address these issues by requiring additional rules and lexical features. Note that in the deterministic approach, it is essential for lexical items to be properly disambiguated or backtracking will be required.

In testing CDP, normal sentences are presented, except that selected words are coded ambiguously (here indicated by angle brackets < > around the word). In the cases shown, the lexically ambiguous words were correctly interpreted and reasonable structures resulted, although lower strengths were observed. CDP utilizes syntactic context to resolve these ambiguities and automatically works to relate novel situations to training cases through the generalization capability of the network. As before, no additional rules or mechanisms are required to provide this capability.

Our examples are all based on examples from Milne (1986). Sentence (1c) contains the word *will* coded ambiguously as an NP and an auxiliary, modal verb. In the context of the sentence, it is clearly being used as a modal auxiliary and the parser treats it that way. A similar result was obtained for sentence (2c) which parses as (29a). In sentence (3c), *hit* is coded to be ambiguous between an NP (as in playing cards) and a verb. The network correctly identifies it as the main verb of the sentence as in sentence (28a). Sentence (4c) is constructed as for sentence (30a). Sentence (5c) presents *can* ambiguously as an auxiliary, modal, and main verb, while *fish* is presented uniquely as an NP. *Can* is processed as the main verb of the sentence and results in the same structure as sentence (31a). Likewise, sentence (6c), which contains *fish* coded ambiguously as a verb/NP and *can* coded uniquely as an auxiliary verb, produces the same structure as sentence (32a). In the cases shown, the lexically ambiguous words were disambiguated and reasonable structures resulted. Note that the overall average strengths were lower than comparable grammatical sentences discussed, as expected.

6. Summary

CDP implements a deterministic parser based on the rules from a deterministic grammar. The result is a combined symbolic/sub-symbolic system which exhibits characteristics from several well-known extensions to the basic deterministic parser. These extended properties come essentially for free due to the use of connectionism. The grammar used in these experiments is derived, with minor modifications, from one used by Marcus, but with much inspiration from Berwick, Charniak, and Milne. Only small modifications are required in the grammar to accommodate our particular architecture in CDP.

7. Acknowledgments

The first author gratefully acknowledges support from King Fahd University. Support from the Center for Intelligent Computer Systems at Washington University is also acknowledged with gratitude. The authors also express gratitude to William Ball, Georg Dorffner, Marios Fourakis, David Harker, Dan Kimura, Barry Kalman, Ron Loui, John Merrill, Gadi Pinkas, and Robert Port for thoughtful discussions and comments concerning this work.

References

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

Berwick, R.C. 1979. "Learning Structural Description of Grammar Rules from Examples" *Proceedings of the 6th International Conference on Artificial Intelligence*, Tokyo, Japan

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

Chomsky, N. 1980. *Rules and Representations.* Columbia University Press, New York.

Faisal, K.A. and S.C. Kwasny. 1990. "Deductive and Inductive Learning in a Connectionist Deterministic Parser." *Proceedings of the International Joint Conference on Neural Networks*, FORTHCOMING, Washington, DC, (Jan 15-19).

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., K.A. Faisal, and W.E. Ball. 1990. "Unifying Several Natural Language Systems in a Connectionist Deterministic Parser." *Proceedings of the Western Multi Conference*, FORTHCOMING, San Diego, CA, (Jan 17-19).

Kwasny, S.C., and Faisal, K.A. 1989a. "Connectionism and Determinism in a Rule-Based Natural Language System." Technical Report WUCS-89-31, Department of Computer Science, Washington University, St. Louis, MO.

Kwasny, S.C. and K.A. Faisal. 1989b. "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. 1980. "Treatment of Ungrammatical and Extra-Grammatical Phenomena in Natural Language Understanding Systems." Indiana University Linguistics Club, Bloomington, Indiana. (Nov.).

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.

TABLE 1: Grammatical Sentences Tested

Sentence Form	Avg. Strength
(1a) John should have scheduled the meeting for Tuesday.	56.9
(2a) Scheduled a meeting for Tuesday.	29.4
(3a) Has John scheduled the meeting for Tuesday?	36.8
(4a) John has scheduled the meeting for Tuesday.	55.1
(5a) The meeting has been scheduled for Tuesday.	565.5
(6a) The meeting seems to have been scheduled for Tuesday.	70.8
(7a) The jar seems broken.	5.3
(8a) I persuaded John to do it.	39.7
(9a) I saw him do it.	38.2
(10a) Mary wants John to have a party.	46.5
(11a) Mary wants to have a party.	57.9
(12a) What will the man put in the corner?	376.2
(13a) What will the man put the book in?	23.7
(14a) Who did John see?	427.3
(15a) Who schedule a meeting?	38.3
(16a) Who is scheduling a meeting?	61.0
(17a) What is the man scheduling?	11.5
(18a) What did Bob give Mary?	32.1
(19a) The man who wanted to meet Mary has disappeared.	33.0
(20a) The man who hit Mary with a book has disappeared.	29.2
(21a) The man whom Mary hit with a book has disappeared.	254.5
(22a) I told that boy that boys should do it.	19.9
(23a) That mouse that the cat chased had squeaked.	8.8
(24a) I told Sue you would schedule the meeting.	4.3
(25a) I told the girl that you would schedule the meeting.	5.8
(26a) John is scheduling the meeting for Tuesday.	54.7
(27a) The boy did hit Jack.	137.7
(28a) Tom hit(v) Mary.	29.5
(29a) Tom will(aux) hit(v) Mary.	125.8
(30a) The will(noun) gave the money to Mary.	61.9
(31a) They can(v) fish(np).	30.0
(32a) They can(aux) fish(v).	6.3

TABLE 2: Ungrammatical Sentences Tested

Sentence Form	Avg. Strength
(1b) *John have should scheduled the meeting for Tuesday.	14.4
(2b) *Has John schedule the meeting for Tuesday?	32.3
(3b) *John is schedule the meeting for Tuesday.	9.5
(4b) *John is is scheduling the meeting for Tuesday.	7.2
(5b) *The boy did hitting Jack.	14.8
(6b) * The meeting is been scheduled for Tuesday.	559.6

TABLE 3: Lexically Ambiguous Sentences Tested

Sentence Form	Avg. Strength
(1c) <Will> John schedule the meeting for Tuesday?	5.0
(2c) Tom <will> hit Mary.	29.8
(3c) Tom <hit> Mary.	13.6
(4c) The <will> gave the money to Mary.	16.6
(5c) They <can> fish(np).	20.6
(6c) They can(aux) <fish>.	2.9