AUTOMATIC GRAMMAR ACQUISITION

Scott Miller

College of Computer Science
Northeastern University
Boston, MA 02115

Heidi J. Fox

BBN Systems and Technologies
70 Fawcett St.,
Cambridge, MA 02138

ABSTRACT

We describe a series of three experiments in which supervised learning techniques were used to acquire three different types of grammars for English news stories. The acquired grammar types were: 1) context-free, 2) context-dependent, and 3) probabilistic context-free. Training data were derived from University of Pennsylvania Treebank parses of 50 Wall Street Journal articles. In each case, the system started with essentially no grammatical knowledge, and learned a set of grammar rules exclusively from the training data. Performance for each grammar type was then evaluated on an independent set of test sentences using Parseval, a standard measure of parsing accuracy. These experimental results yield a direct quantitative comparison between each of the three methods.

1. INTRODUCTION

Designing and refining a natural language grammar is a difficult and time-intensive task, often consuming months or even years of skilled effort. The resulting grammar is usually not completely satisfactory, failing to cover a significant fraction of the sentences in the intended domain. Conversely, the grammar is likely to overgenerate, leading to multiple interpretations for a single sentence, many of which are incorrect. With the increasing availability of large, machine-readable, parsed corpora such as the University of Pennsylvania Treebank [Santorini, 90], it has become worthwhile to consider automatic grammar acquisition through the application of machine learning techniques. By learning a grammar that completely covers a training set for some domain, it is hoped that coverage will also be increased for new sentences in that domain. Additionally, machine learning techniques may be useful in reducing overgeneration through a variety of techniques that have been suggested in recent literature. One suggestion is to introduce local contextual information into a grammar [Simmons and Yu, 92], based on the premise that local context provides useful information for selecting among competing grammar rules. A second suggestion is to introduce probabilities in the form of a probabilistic context-free grammar [Chitaro and Grishman, 90], based on the premise that a combination of local probability measures provides a useful estimate of the probability of an entire parse.

In this work, we investigate both of these suggestions and compare them with a simple, automatically learned, context-free grammar. In each case, the grammar is acquired from a subset of parsed Wall Street Journal articles taken from the University of Pennsylvania Treebank. We then apply the acquired grammar to the problem of producing a single unambiguous parse for each sentence of an independent test set derived from the same source.

2. LEARNING A CONTEXT-FREE GRAMMAR

A context-free grammar is acquired from a parsed Treebank corpus by straightforward memorization of the grammar rules used in each training example. Figure 1 shows a typical parse tree from our training corpus; Table 1 shows the grammar rules used.

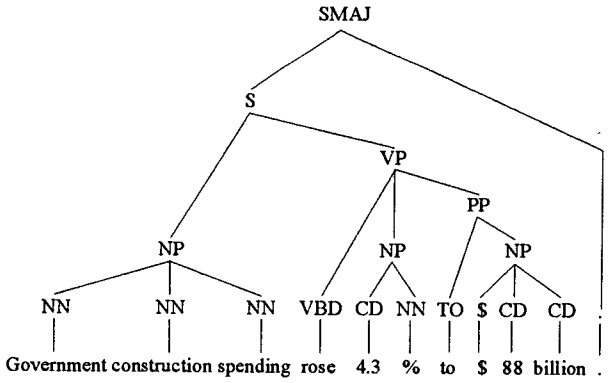

Figure 1: A typical parse tree from the Treebank corpus.

SMAJ → S .	S → NP VP	NP → NN NN NN
VP → VBD NP PP	NP → CD NN	PP → TO NP
NP → $ CD CD		

Table 1: Rules used for the parse tree shown in figure 1.

In order to parse new sentences, a simple bottom-up chart parser is combined with the acquired grammar. In our experiments, the parser is run until a single parse tree is found that spans all of the words in the input sentence. If no such tree is found, then

the minimum number of disjoint fragments is returned such that the set of fragments spans the entire input sentence. Because the acquired grammar is highly ambiguous, the returned parse tree is dependent on the order in which the grammar rules are applied. To account for this sensitivity to rule order, we repeat our experiments several times with different rule orderings. We have found that, although different orderings produce different parse trees, the overall accuracy of the results do not differ significantly. As expected, the high degree of rule ambiguity, together with our procedure that returns a single parse tree for each sentence, yields rather poor performance. Nevertheless, the performance of this system serves as a baseline which we use to assess the performance of other systems based on alternative grammar types.

3. LEARNING A CONTEXT-DEPENDENT GRAMMAR

In this experiment, we closely follow the approach of Simmons and Yu, with extensions to accommodate grammar rules of a form derivable from the Treebank. Unlike our other experiments, the grammar rules in this experiment are situation / action rules for a shift-reduce parser. In the following sections we consider:

- The general structure of the shift-reduce parser.
- The form of the context-dependent rules.
- The problem of learning context-dependent rules for a shift-reduce parser from Treebank examples.
- A parsing strategy that attempts to find a single best parse based on contextual information.

3.1. Shift-Reduce Parser

The shift-reduce parser consists of two primary data structures: a five position input buffer, and an unlimited depth push down stack. New words arriving at the parser flow, in the order in which they are received, through the input buffer. Shift operations remove the leading word from the input buffer and push it onto the top of the stack.

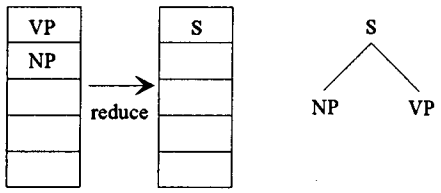

Figure 2: Reduce operations construct tree structures.

When this occurs, all other words are shifted one position toward the front of the buffer, and the next unprocessed word is moved into the last buffer position. Reduction operations remove two or more elements from the top of the stack, and replace them with a single constituent. Reduction operations are equivalent to constructing parse subtrees, as in Figure 2.

3.2. Context-dependent Rules

The determination of what action the parser takes in a particular situation is governed by context-dependent rules. Constraints given by the rules are matched against actual situations in a two part process. First, for a rule to be applicable, a hard constraint specified by two or more elements on the top of the stack must be satisfied. Next, those rules that satisfy this condition are ordered by preference based on soft constraints specified by context elements of the stack and buffer.

Hard constraints for reduction rules are determined directly by the reductions themselves. For example, to apply a rule reducing {DT JJ NN ...} to {NP ...}, the top three stack elements must be NN, JJ, and DT. For shift operations, the hard constraints are always given by the top two stack elements.

Soft constraints are specified by a two part context comprised of a stack portion and a buffer portion.. The stack portion is comprised of the three stack positions directly below the hard constraint, while the buffer portion is comprised of the entire five element buffer. Soft constraints are scored by a weighted sum of the number of matches between rule and situation contexts. These weights were hand tuned to maximize parsing accuracy.

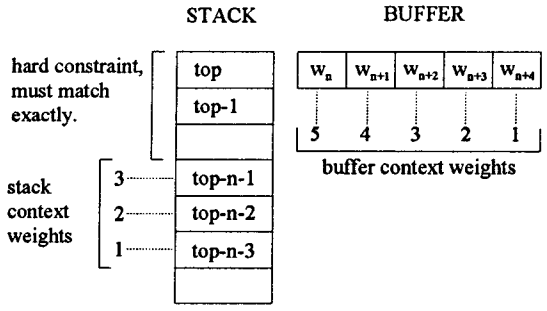

Figure 3: The primary data structures of the shift-reduce parser.

3.3. Learning Shift-Reduce Rules

In order to train the shift-reduce parser, it is first necessary to convert the Treebank parse trees into sequences of shift and reduce operations. A simple treewalk algorithm that performs the conversion is shown in Figure 4.

Training examples are presented successively to the parser. For each example, all rules with satisfied hard constraints are

formed into a list. Next, a shorter list is formed by extracting only those rules that best satisfy the soft constraints. If the correct parser action is among those specified by the shortened list of rules, then no action is taken. Otherwise, a new rule is formed from the current context and parser action, and is stored in the hash table of rules. When training is complete, the rule matching mechanism can present a short list of possible rules, one of which is guaranteed to be correct, for every situation presented in the training examples.

```
Convert(tree)
BEGIN
  IF tree is a leaf node THEN
    emit a Shift operation
  ELSE {has child nodes}
    FOR ALL child nodes of tree (from left to right) DO
      Convert(child node)
    END FOR
    emit a Reduce operation
    {reducing child nodes to a single symbol}
  END IF
END
```

Figure 4: An algorithm for converting a parse tree into shift-reduce parser actions.

3.4. Shift-Reduce Parser Operation

Parsing a sentence is considered as a search problem, the goal of which is to find a sequence of actions that lead from an initial parser state to a final state. The initial state for a sentence is characterized by an empty stack and a buffer filled with the first five words of the sentence. The final state is characterized by an empty buffer and a single element at the top of the stack. Valid transitions between states are determined by the rules acquired during training.

Given a parser state, the rule matching mechanism returns a list of rules specifying actions that cause transitions to new states. The rules are guaranteed to be legal by hard constraints, but vary to the degree to which soft constraints on context are satisfied. Each alternative rule corresponds to a different syntactic interpretation of a phrase, only one of which is correct. The premise, put forth by Simmons and Yu, is that the use of context information significantly reduces ambiguity.

To parse a sentence, a beam search is used to find a path through the state space that maximizes the soft constraints specifying context. Upon completion, a list of shift and reduce operations is returned. These operations correspond directly to a parse tree for the input sentence.

4. LEARNING A PROBABILISTIC CONTEXT-FREE GRAMMAR

In this experiment, probabilities are used to select one among the set of alternative parse trees derivable for an input sentence. A straightforward evaluation of the probability of a parse tree is obtained from the probabilities of the individual grammar rules that comprise the tree. For each rule r of the form $\alpha \rightarrow \beta$, the rule probability is given by $P(r) = P(\beta|\alpha)$. Then, given a parse tree t constructed according to a derivation $D(t)$, the probability of t is the product of all the conditional rule probabilities in the derivation:

$$P(t) = \prod_{r \in D(t)} P(r).$$

Using the Treebank training corpus, $P(\beta|\alpha)$ is estimated by counting the number of times the rule $\alpha \rightarrow \beta$ appears in the training, divided by the number of times the nonterminal symbol α appears. In order to parse new sentences, a simple bottom-up chart parser is extended to include probability measures. In particular, an extra field is added to the chart structure in order to store a probability value corresponding to each completed edge. When multiple derivations result in the same edge, the probability value stored is the maximum among the competing theories. When parsing a sentence, all possible derivations are considered, and the derivation with the highest probability is then returned.

5. EXPERIMENTAL RESULTS

Each of the grammars was learned from a training set of 731 sentences (16,733 words) from the Wall Street Journal Treebank corpus. A separate test set of 49 sentences (1289 words) was compiled from the same corpus. Parse quality was evaluated using Parseval, which reports three different measures of correctness: recall, precision, and crossings. Each parse tree to be evaluated (the candidate parse) is compared against the corresponding parse as found in Treebank (the standard parse). Recall measures the percentage of the constituents in the standard parse which are present in the candidate parse. Precision measures the percentage of the constituents in the candidate parse which are correct (i.e., present in the standard parse). Crossings measures the number of constituents in the candidate parse which are incompatible with the constituents in the standard parse, where incompatibility means that the constituent crosses brackets with a constituent in the standard. For more details on the evaluation procedure, see [Black, *et al.*, 91]

The results of the test are shown in Table 2. As expected, the performance of the simple context-free grammar is substantially worse than the performance of both the context-dependent grammar and the probabilistic context-free grammar. It is interesting to note that although recall for the P-CFG and CDG is essentially equal, the P-CFG has a higher precision. This

suggests that probabilistic modeling is more successful at reducing overgeneration than simple examination of context. The P-CFG also shows a lesser average number of crossings per sentence.

	Crossings	Recall	Precision
P-CFG	4.94	52.75	51.52
CDG	6.61	51.20	42.16
CFG	11.06	28.49	22.25

Table 2: Parseval results for each grammar

6. CONCLUSIONS

These experiments provide a quantitative measure of the relative effectiveness of the three different types of grammars. Using the standard context-free grammar as a baseline, we see great improvement both with the addition of context information and with the incorporation of a probabilistic model. We also see evidence that using context to disambiguate among rules is not as effective as using probabilities.

There are still many problems to overcome. Direct conversion of Treebank parse trees into rules yields productions whose right-hand sides can vary in size between 1 and approximately 10. This is suspected to have significant impact on the performance of the context-dependent system.

More improvements will be necessary before a trainable parser will be able to produce parses of high enough quality to be useful in an understanding system. This increase in accuracy should be achievable by combining the strengths of the context-dependent model with those of the probabilistic context-free model, and by exploring ways to make use of other types of information, such as semantic information. It would also be worthwhile to further experiment with varying the amount of training data, contrasting domain-dependent and domain-independent training, and varying the amount and type of context information used by the context-dependent model.

ACKNOWLEDGEMENTS

The work reported here was supported in part by the Defense Advanced Research Projects Agency and was monitored by the Rome Air Development Center under Contract No. F30602-91-C-0051. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

REFERENCES

1. Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., Strzalkowski, T., *A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars: Proceedings of the Speech and Natural Language Workshop,* Morgan Kaufmann Publishers, pages 306-311, February 1991.

2. Chitaro, M.V. and Grishman, R., *Statistical Parsing of Messages: Proceedings of the Speech and Natural Language Workshop,* Morgan Kaufmann Publishers, pages 263-276, June 1990.

3. Santorini, B., *Annotation Manual for the Penn Treebank Project*, Technical Report, CIS Department, University of Pennsylvania, May 1990.

4. Simmons, R. and Yu Y., *The Acquisition and Use of Context-Dependent Grammars for English: Computational Linguistics*, Vol. 18, Number 4, pages 391-416, Dec. 1992.