A Corpus-based Probabilistic Grammar with Only Two Non-terminals

Satoshi SEKINE **Ralph GRISHMAN**

Computer Science Department
New York University
715 Broadway, 7th floor
New York, NY 10003, USA
sekine,grishman@cs.nyu.edu

Abstract

The availability of large, syntactically-bracketed corpora such as the Penn Tree Bank affords us the opportunity to automatically build or train broad-coverage grammars, and in particular to train probabilistic grammars. A number of recent parsing experiments have also indicated that grammars whose production probabilities are dependent on the context can be more effective than context-free grammars in selecting a correct parse.

To make maximal use of context, we have automatically constructed, from the Penn Tree Bank version 2, a grammar in which the symbols `S` and `NP` are the only real non-terminals, and the other non-terminals or grammatical nodes are in effect embedded into the right-hand-sides of the `S` and `NP` rules. For example, one of the rules extracted from the tree bank would be `S -> NP VBX JJ CC VBX NP [1]` (where NP is a non-terminal and the other symbols are terminals – part-of-speech tags of the Tree Bank). The most common structure in the Tree Bank associated with this expansion is `(S NP (VP (VP VBX (ADJ JJ) CC (VP VBX NP)))) [2]`. So if our parser uses rule [1] in parsing a sentence, it will generate structure [2] for the corresponding part of the sentence.

Using 94% of the Penn Tree Bank for training, we extracted 32,296 distinct rules (23,386 for S, and 8,910 for NP). We also built a smaller version of the grammar based on higher frequency patterns for use as a back-up when the larger grammar is unable to produce a parse due to memory limitation. We applied this parser to 1,989 Wall Street Journal sentences (separate from the training set and with no limit on sentence length). Of the parsed sentences (1,899), the percentage of no-crossing sentences is 33.9%, and Parseval recall and precision are 73.43% and 72.61%.

1 Introduction

The availability of large, syntactically-bracketed corpora such as the University of Pennsylvania Tree Bank affords us the opportunity to automatically build or train broad-coverage grammars. Although it is inevitable that a structured corpus will contains errors, statistical methods and the size of the corpus may be able to ameliorate the effect of individual errors. Also, because a large corpus will include examples of many rare constructs, we have the potential of obtaining broader coverage than we might with a hand-constructed grammar. Furthermore, experiments over the past few years have shown the benefits of using probabilistic information in parsing, and the large corpus allows us to train the probabilities of a grammar [8] [7] [11] [2] [4] [12].

A number of recent parsing experiments have also indicated that grammars whose production probabilities are dependent on the context can be more effective than context-free grammars in selecting a correct parse. This context sensitivity can be acquired easily using a large corpus, whereas human ability to compute such information is obviously limited. There have been

several attempts to build context-dependent grammars based on large corpora. [14] [11] [13] [2] [4] [12].

As is evident from the two lists of citations, there has been considerable research involving both probabilistic grammar based on syntactically-bracketed corpora and context-sensitivity. For example, Black proposed 'History based parsing', a context-dependent grammar trained using a large corpus [2]. History-based parsing uses decision-tree methods to identify the most useful information in the prior context for selecting the next production to try. This approach, however, requires a hand-constructed grammar as a starting point.

Bod [4] has described how to parse directly from a tree bank. A new parse tree can be assembled from arbitrary subtrees drawn from the tree bank; the parser searches for the combination with highest probability. This can, in principle, take advantage of arbitrary context information. However, the search space is extremely large, so a full search is not practical, even for a small tree bank (Bod proposes using Monte Carlo methods instead). Results have been reported only for a small tree bank of 750 ATIS sentences.

In this paper we will present a parsing method which involves both probabilistic techniques based on a syntactically-bracketed corpus and context sensitivity. We will describe a very simple approach which allows us to create an efficient parser and to make use of a very large tree bank.

2 An "Ultimate Parser" and a Compromise

An "Ultimate parser" : parsing by look-up

Because of the existence of large syntactically-bracketed corpora and the advantage of context-sensitive parsing, we can contemplate an ultimate parsing strategy - parsing by table look-up. This approach is based on the assumption that the corpus covers most of the possible sentence structures in a domain. In other words, most of the time, you can find the structure of any given sentence in the corpus. If this assumption were correct, we could parse a sentence just by look-up. The system first assigns parts-of-speech to an input sentence using a tagger, and then just searches for the same sequence of parts-of-speech in the corpus. The structure of the matched sequence is the output of the system. Now we will see if the assumption is correct. We investigated the Penn Tree Bank corpus version 2, which is one of the largest syntactically-bracketed corpora, but it turned out that this strategy does not look practical. Out of 4,7219 sentences in the corpus[1], only 2,232 sentences (4.7%) have exactly the same structure as another sentence in the corpus. This suggests that, if we apply the above strategy, we would find a match, and hence be able to produce a parse for only 4.7% of the sentences in a new text.

Compromise

We therefore have to make a compromise. Instead of seeking a complete match for the part-of-speech sequence of an entire sentence, we introduce partial sequence matchings based on the two important non-terminals in the Penn Tree Bank, `S` (sentence) and `NP` (noun phrase). We try to find a nested set of `S` and `NP` fragments in the given sentence so that the whole sentence is derived from a single `S` and then apply the look-up strategy for each fragment. In other words, at first the system collects, for each instance of `S` and `NP` in the training corpus, its expansion into `S`'s, `NP`'s, and lexical categories; this is, in effect, a production in a grammar with non-terminals `S` and `NP`. It also records the full constituent structure for each instance. In analyzing a new sentence, it tries to find the best segmentation of the input into `S`'s and `NP`'s; it then outputs the combination of the structures of the chosen segments. To assure that this strategy is applicable, we collected statistics from the Penn Tree Bank (Table 1). From the table, we can see that there are a considerable number of multiple occurrences of structures, and that a very small number of structures covers a large number of instances in the corpus. The most frequent structures for `S` and `NP` are shown just below. The numbers on the left indicate their frequency. Most of

[1] We used 96% of the corpus which will be used for the grammar training. We also processed minor modifications 1-3, which will be described later.

Category	S	NP
Total instances	88,921	351,113
Distinct structures	24,465	9,711
Number of structures which cover 50% of instances	114	7
percentage of instances covered by structures of 2 or more occurrences	77.2%	98.1%
percentage of instances covered by top 10 structures	27.5%	57.9%

Table 1: Statistics of S and NP structures

the symbols are from Penn Tree Bank[2]; symbols which we have introduced (for example, NNX or VBX) are explained later.

```
6483 (S NP (VP VBX NP))                    36470 (NP DT NNX)
4931 (S -NONE-)                            34408 (NP NP (PP IN NP)
4188 (S NP (VP VBX (SBAR S)))              32641 (NP NNPX)
1724 (S NP (VP VBG NP)                     27432 (NP NNX)
1549 (S S , CC S)                          17731 (NP PRP)
```

Although we see that many structures are covered by the data in the corpus, there could be ambiguities where two or more structures can be created from an identical part-of-speech sequence. For example, the prepositional attachment problem leads to such ambiguities. The survey on Penn Tree Bank shows us the percentage of the sequences which could be derived from S and NP with different structures are 7% and 12%, respectively. The maximum number of different structures for the same part-of-speech sequence are 7 for S and 12 for NP. However, by taking the most frequent structure for each ambiguous sequence, we can keep such mistakes to a minimum. We find that the errors caused by this are 8% and 3% for S and NP, respectively. We believe these errors can be reduced by introducing lexical or semantic information in the parsing. This will be discussed later.

From these statistics, we can conclude that many structures of S and NP can be covered by the data in the Penn Tree Bank. This result supports the idea of the parsing with two non-terminals, S and NP which segment the input, and the structure inside the segment is basically decided by table look-up. However, because we introduce non-terminals and hence introduce ambiguities of segment boundary, the overall process becomes more like parsing rather than just table look-up.

3 Grammar

Guided by the considerations in the last section, we try to build a grammar automatically from the Penn Tree Bank. The grammar has symbols S and NP as the only non-terminals, and the other non-terminals or grammatical nodes in the Tree Bank are in effect embedded into the right-hand-sides of the S and NP rules. For example, the following is one of the extracted rules.

```
S -> NP VBX JJ CC VBX NP
    :structure "(S <1> (VP (VP <2> (ADJ <3>)) <4> (VP <5> <6>)))";
```

(where S and NP are non-terminals and the other symbols in the rules are terminals – part-of-speech tags of the Penn Tree Bank). By this rule, S is replaced by the sequence NP VBX JJ CC

[2]Some category symbols defined in the Penn Tree Bank: VBP: non-3rd singular present-tense verb, VBZ: 3rd-person singular present-tense verb, VBD: past-tense verb, VBG: present-particle verb, NNP: singular proper noun, NNPS: plural proper noun , NN: singular or mass noun, NNS: plural noun, CD: cardinal number, FW: foreign word, SYM: symbol, CC: coordinating conjunction, IN: preposition and subordinate conjunction.

VBX NP, and in addition the rule creates a tree with grammatical non-terminals, VP's and ADJ. When the parser uses the rule in parsing a sentence, it will generate the associated structure. For example, Figure 1 shows how the sentence "This apple pie looks good and is a real treat" is parsed. The first three words and the last three words in the sentence are parsed as

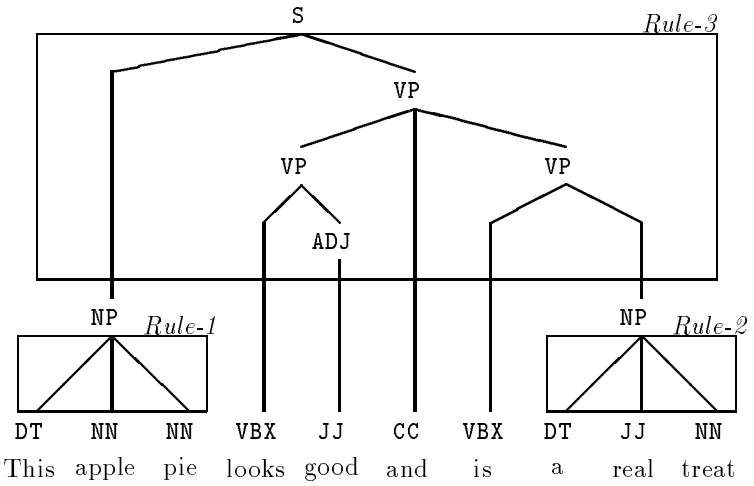

Figure 1: Example of parsed tree

usual, using the rules, NP -> DT NN NN (*Rule-1*) and NP -> DT JJ NN (*Rule-2*), respectively. The remainder is parsed by the single rule, S -> NP VBX JJ CC VBX NP (*Rule-3*). This rule constructs the entire structure under the root S. The whole tree is generated based on the three rules, although there are more than three grammatical non-terminals in the tree.

Minor modification

We made four kinds of minor modification to the grammar, in order to improve its coverage and accuracy. First, the punctuation mark at the end of each sentence is deleted. This is to keep consistency at the end of sentences, which sometimes have a period, another symbol or no punctuation. Second, similar categories, in terms of grammatical behavior, are merged into a single category. This reduces the number of grammar rules and increases coverage of the grammar. For example, present tense and past tense verbs play a similar role in determining grammatical structure. Third, sequences of particular categories are collapsed into single instances of the category. For example, sequences of numbers, proper nouns or symbols are replaced automatically by a single instance of number, proper noun or symbol. This modification also works to reduce the number of grammar rules. Finally, we know that the Penn Tree Bank project tried to reduce the number of part-of-speech categories in order to ease the tagging effort. The Penn Tree Bank manual [10] says that they combine categories, in cases where finer distinctions can be recovered based on lexical information. So, by introducing new categories for a set of words which have different behavior from the other words in the same category, we can expect to get more information and more accurate parses.

The following is the list of modifications in the grammar:

1. Delete punctuation at the end of sentences

2. Merge Categories
   VBX=(VBP, VBZ, VBD), NNPX=(NNP, NNPS), NNX=(NN, NNS)

3. Collapse sequence into single instance
   `NNP, CD, FW, SYM`

4. Introduce new categories
   `@OF = of`;
   `@SNC =` Subordinating conjunction which introduce sentence (`although`, `because`, `if`, `once`, `that`, `though`, `unless`, `whether`, `while`);
   `@DLQ =` Pre-quantifier adverbs (`about`, `all`, `any`, `approximately`, `around`, `below`, `even`, `first`, `just`, `next`, `not`, `only`, `over`, `pretty`, `second`, `so`, `some`, `too`)

Tagging

As the first step in parsing a sentence, one or more part-of-speech tags are assigned to each input token, based on the tags assigned to the same token in the training corpus. (Note that this introduces ambiguity.) Each tag has a probability which will be used in the score calculation in parsing. The probability is based on the relative frequency of the tag assigned to the token in the corpus. We set the threshold for the probability to 5% in order to make the parser efficient. Tags with smaller probability than the threshold are discarded.

$$P_{tag}(t|w) = \frac{Frequency\ of\ word\ w\ with\ tag\ t}{Frequency\ of\ word\ w} \tag{1}$$

Score Calculation

The formulae for the probability of an individual rule P_{rule} and the score of a parsed tree S_{tree} are the following. The probability of a rule, $X->Y$, where Y is a sequence, is based on the frequency with which X dominates Y in the corpus, and the frequency of the non-terminal, X. The score for a parse tree is the product of probability of each rule used to build the tree together with square of probability of the tag for each word in the sentence. The square factor results in putting more weight on tag-probability over rule-probability, which produce better results than balancing the weights. The best parsed tree has the highest score among the trees possibly derived from the input.

$$P_{rule}(X->Y) = \frac{Frequency\ with\ which\ X\ is\ expanded\ as\ Y}{Frequency\ of\ X} \tag{2}$$

$$S_{tree}(T) = \prod_{R:\ rules\ in\ T} P_{rule}(R) \prod_{t:\ tags\ in\ T} (P_{tag}(t|w))^2 \tag{3}$$

Backup Grammar

Although we built a parser which can handle a large grammar, as described in the next section, it is unable to parse some long sentences, because of computer memory limitations. So we prepared a smaller grammar, for use in case the larger grammar can't parse a sentence. The small grammar consists of the rules having frequency more than 2 in the corpus. Because the number of rules is small, parsing is rarely blocked by space limitations. The parsed result of this grammar is used only when the larger grammar does not produce a parse tree. Table 2 shows the numbers of rules in the larger grammar(G-0) and the smaller grammar(G-2). The number of rules in G-0 is smaller than the number of 'distinct structures' shown in Table 1, because if there are several structures associated with one sequence, only the most common structure is kept.

4 Parser

It is not easy to handle such a large grammar. For example, a simple LR parser would need a huge table, while a simple chart parser would need a large number of active nodes. We therefore

Category	Grammar-0	Grammar-2
S	23,386	2,478
NP	8,910	2,087
Total	32,296	4,565

Table 2: Number of rules

developed a chart parser which can handle a large grammar. The key technique is that it factors grammar rules with common prefixes. Actually, the grammar rules are stored like a finite state automaton. As the grammar has thousands of rules which start from, for instance, DT, a simple chart parser has to have that same number of active nodes when it find a determiner in the input sentence. However, since active nodes indicate grammar rules which can be extended after that point, we can replace the thousands of nodes by a single pointer which points to the corresponding node in the grammar automaton. Because this node has an arc to all the possible successors, it is equivalent to thousands of active nodes in a conventional chart. Also, because we try to find only the best (highest possibility) parse tree, we can eliminate inactive nodes whose score is lower than another inactive node of the same category and span. The limits for active nodes and inactive nodes are set to 3,000,000 and 10,000, because of memory limitations.

5 Experiment

For our experiments, the WSJ corpus of the Penn Tree Bank is divided into two portions. One is used for training (96%) and the other part is used for testing. The training corpus is used to extract grammar rules and the test corpus contains 1,989 sentences. The parsing results are shown in Table 3. Here, "G-0" is the parsed result using the grammar with all the produced

Grammar	number of sentence	no parse	space exhausted	sentence length	run time (sec./sent.)
G-0	1989	20	293	19.9	13.6
G-2	1989	172	42	22.2	8.1

Table 3: Parsing Result

rules, "G-2" is the grammar with rules of frequency more than 2. "No parse" means the parser can't get S for the whole structure, "space exhausted" means that the node space of the parser is exhausted in the middle of the parsing. "Sentence length" is the average length of parsed sentences, and the run time is the parse time per sentence in seconds using a SPARC 5. Although the average run time is quite high, more than half of the sentences can be parsed in less than 3 seconds while a small number of sentences take more than 60 seconds. We can see that the number of "no parse" sentences with G-2 is larger than that with G-0. This is because there are fewer grammar rules in G-2, so some of the sequences have no matching pattern in the grammar. It is natural that the number of sentences which exhaust memory is larger for G-0 than for G-2, because of the larger number of rules.

Next, the evaluation using parseval method on parsed sentences is shown in Table 4. "Parseval" is the measurement of parsed result proposed by Black et.al. [1]. The result in the table is the result achieved by the G-0 grammar, supplemented by the result using the G-2 grammar, if the larger grammar can't generate a parse tree. These numbers are based only on the sentences which parsed (1,899 out of 1,989 sentences; in other words 90 sentences are left as unable-to-be-parsed sentences even using the back-up method). Here, "complete match" means that the result is exactly the same as the corresponding tree in the Penn Tree Bank. "No-crossing" is the

Total sentences	1899
No-crossing	643 (33.9%)
Ave.crossing	2.64
Parseval (recall)	73.43%
Parseval (precision)	72.61%

Table 4: Evaluation Result

number of sentences which have no crossing brackets between the result and the corresponding tree in the Penn Tree Bank. "Ave.crossing" is the average number of crossings per sentence.

It is not easy to compare these numbers between systems, because some conditions of the test sentences, length, complexity, etc., affect the result. However, roughly speaking, these numbers are comparable to or better than the score of so-called traditional grammar, or hand-made grammars. For example, Black [3] cited the best non-crossing score using a traditional grammars as 41% and the average of several systems as 22%.

6 Future work

Analyzing the errors made by the parser, we found that a considerable number of unparsed sentences (including no-parse and memory-exhausted sentences) and wrongly parsed sentences contain special symbols, like ':', '–', '(' or ')'. Our strategy to enumerate structures, is not good at parsing sentences which have rare tokens, like these symbols. Furthermore, there are some odd sentences involving these symbols. For example, there are sentences which have an unbalanced parenthesis, because of incorrect division of the text into sentences or multiple sentences within a single pair of parentheses. In order to parse these sentences, we believe, special pre-treatments are needed.

As was mentioned earlier, there are several part-of-speech sequences which could generate several different structures. One source of ambiguity is annotator's inconsistencies, which we can't deal with. Another major source is attachment problems, such as prepositional attachment or conjunctive (and, or) attachment. Although it is well known that some of these ambiguities are unsolvable using only the context within the sentence, many of them are heavily related to the lexical or semantic information within the sentence. We have been conducting research on automatically acquiring these selectional constraints, and we are planning to incorporate this semantic knowledge into the parser [9], [15].

Introducing lexical information in the parser is also useful for other kinds of the structural disambiguation. We showed this in minor modification 4, by introducing new categories based on lexicon. However, the method we used to choose the candidates depended on human intuition. We are considering creating an automatic method to identify those words for which it is beneficial to make a new category.

7 Conclusion

We developed a corpus-based probabilistic grammar whose rules are extracted from syntactically-bracketed corpora. The grammar has only two non-terminals, `S` and `NP`. The rules of the grammar contain grammatical non-terminal within the tree. This feature introduces context-sensitivity for the terminals.

Corpus-based grammar generation has a significant advantage over building a grammar by hand, particularly if we aspire to a high degree of coverage. Once we have a syntactically-bracketed corpus for a new domain, a grammar can be automatically created for that domain. While building syntactically-bracketed corpora is not so easy, large-scale corpora have been

successfully constructed by teams of coders; building a very-broad-coverage grammar by hand has proven much more challenging.

We conducted an experiment using two grammars derived from a tagged corpus. The accuracy is about the same or better than with a conventional grammar. As this grammar uses only part-of-speech information, we may be able to improve it by incorporating lexical or semantic information.

8 Acknowledgments

The work reported here was supported by the Advanced Research Projects Agency under contract DABT63-93-C-0058 from the Department of the Army. Also, we would like to thank our colleagues at NYU and the member of Penn Tree Bank Project.

References

[1] E.Black, et.al. "A procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars" *Proc. of Fourth DARPA Speech and Natural Language Workshop* (1991)

[2] E.Black, F.Jelinek, J.Lafferty, D.Magerman, R.Mercer, S.Roukos "Towards History-based Grammars: Using Richer Models for Probabilistic Parsing" *ACL-93* (1993)

[3] E.Black "Parsing English By Computer: The State Of the Art" *ATR International Workshop on Speech Translation* (1993)

[4] R.Bod "Using an Annotated Corpus as a Stochastic Grammar" *EACL-93* (1993)

[5] E.Brill "Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach" *ACL-93* (1993)

[6] T.Briscoe, J.Carroll "Generalized Probabilistic LR Parsing of Natural Language (corpora) with Unification-Based Grammars" *Computational Linguistics Vol.19,No.1* (1993)

[7] M.Chitaro, R.Grishman "Statistical parsing of messages" *In proceedings of the Speech and Natural Language Workshop* (1990)

[8] R.Garside, F.Leech "A Probabilistic Parser" *EACL-85* (1985)

[9] R.Grishman, J.Sterling "Generalizing Automatically Generated Selectional Patterns" *COLING-94* (1994)

[10] M.Marcus, B.Santorini, M.Marcinkiewicz "Building a large annotated corpus of English: the Penn Tree bank" *in the distributed Penn Tree Bank Project CD-ROM*, Linguistic Data Consortium, University of Pennsylvania.

[11] D.Magerman, C.Weir "Efficiency, Robustness and Accuracy in Picky Chart Parsing" *ACL-92* (1992)

[12] D.Magerman "Statistical Decision-Tree Models for Parsing" *ACL-95* (1995)

[13] Y.Shabes, R.Waters "Lexicalized Context-Free Grammars" *ACL-93* (1993)

[14] R.Simmons, Y.Yu "The Acquisition and Application of Context Sensitive Grammar for English" *ACL-91* (1991)

[15] S.Sekine, S.Ananiadou, J.J.Carroll, J.Tsujii: "Linguistic Knowledge Generator" *COLING-92* (1992)