Edge-Based Best-First Chart Parsing *

Eugene Charniak and **Sharon Goldwater** and **Mark Johnson**
(ec/sjg/mj@cs.brown.edu)
Departments of Computer Science/Cognitive and Linguistic Sciences
Brown University
Providence RI 02912

Abstract

Best-first probabilistic chart parsing attempts to parse efficiently by working on edges that are judged "best" by some probabilistic figure of merit (FOM). Recent work has used probabilistic context-free grammars (PCFGs) to assign probabilities to constituents, and to use these probabilities as the starting point for the FOM. This paper extends this approach to using a probabilistic FOM to judge edges (incomplete constituents), thereby giving a much finer-grained control over parsing effort. We show how this can be accomplished in a particularly simple way using the common idea of binarizing the PCFG. The results obtained are about a factor of twenty improvement over the best prior results — that is, our parser achieves equivalent results using one twentieth the number of edges. Furthermore we show that this improvement is obtained with parsing precision and recall levels superior to those achieved by exhaustive parsing.

1 Introduction

Finding one (or all) parses for a sentence according to a context-free grammar requires search. Fortunately, there are well known $O(n^3)$ algorithms for parsing, where n is the length of the sentence. Unfortunately, for large grammars (such as the PCFG induced from the Penn II WSJ corpus, which contains around $1.6 \cdot 10^4$ rules) and longish sentences (say, 40 words and punctuation), even $O(n^3)$ looks pretty bleak.

One well-known $O(n^3)$ parsing method (Kay, 1980) is chart parsing. In this approach one maintains an agenda of items remaining to be processed, one of which is processed during each iteration. As each item is pulled off the agenda, it is added to the chart (unless it is already there, in which case it can be discarded) and used to extend and create additional items. In "exhaustive" chart parsing one removes items from the agenda in some relatively simple way (last-in, first-out is common), and continues to do so until nothing remains.

A commonly discussed alternative is to remove the constituents from the agenda according to a *figure of merit* (*FOM*). The idea is that the FOM selects "good" items to be processed, leaving the "bad" ones— the ones that are not, in fact, part of the correct parse— sitting on the agenda. When one has a completed parse, or perhaps several possible parses, one simply stops parsing, leaving items remaining on the agenda. The time that would have been spent processing these remaining items is time saved, and thus time earned.

In our work we have found that exhaustively parsing maximum-40-word sentences from the Penn II treebank requires an average of about 1.2 million edges per sentence. Numbers like this suggest that any approach that offers the possibility of reducing the work load is well worth pursuing, a fact that has been noted by several researchers. Early on, Kay (1980) suggested the use of the chart agenda for this purpose. More recently, the statistical approach to language processing and the use of probabilistic context-free grammars (PCFGs) has suggested using the PCFG probabilities to create a FOM. Bobrow (1990) and Chitrao and Grishman (1990) introduced best-first PCFG parsing, the approach taken here. Subsequent work has suggested different FOMs built from PCFG probabilities (Miller and Fox, 1994; Kochman and Kupin, 1991; Magerman and

* This material is based on work supported in part by NSF grants IRI-9319516 and SBR-9720368, and by ONR grant N0014-96-1–0549.

Marcus, 1991).

Probably the most extensive comparison of possible metrics for best-first PCFG parsing is that of Caraballo and Charniak (henceforth C&C) (Forthcoming). They consider a large number of FOMs, and view them as approximations of some "ideal" (but only computable after the fact) FOM. Of these they recommend one as the best of the lot. In this paper we basically adopt both their framework and their recommended FOM. The next section describes their work in more detail.

Besides C&C the work that is most directly comparable to ours is that of Goodman (1997) and Ratnaparki (1997). Goodman uses an FOM that is similar to that of C&C but one that should, in general, be somewhat more accurate. However, both Goodman's and Ratnaparki's work assumes that one is doing a beam search of some sort, rather than a best-first search, and their FOM are unfortunately tied to their frameworks and thus cannot be adopted here. We briefly compare our results to theirs in Section 5.

As noted, our paper takes off from that of C&C and uses the same FOM. The major difference is simply that our parser uses the FOM to rank edges (including incomplete edges), rather than simply completed constituents, as was done by C&C. What is interesting about our approach is that such a seemingly simple change can produce rather dramatic results. Rather than the thousands of edges required by C&C, the parser presented here requires hundreds, or even, if one is willing to pay a small price in accuracy, tens.

2 Constituent-Based Best-First Chart Parsing

In the approach taken in C&C, only completed edges, i.e., constituents, are entered into the agenda; incomplete edges are always processed as soon as they are constructed. At each iteration the constituent with the highest figure of merit is removed from the agenda, added to the chart, and used to extend current partially completed constituents. Thus we characterize their work as *constituent-based best-first chart parsing*.

C&C take as an "ideal" FOM the quantity $p(N^i_{j,k} \mid t_{0,n})$. Here $N^i_{j,k}$ is a constituent of type i (e.g., NP, VP, etc.) that spans the constituents from j up to but not including k, and $t_{0,n}$ are the n parts-of-speech (tags) of the sentence. Note that C&C simplify parsing by assuming that the input is a sequence of tags, not words. We make the same assumption in this paper. Thus taking $p(N^i_{j,k} \mid t_{0,n})$ as an FOM says that one should work on the constituent that is most likely to be correct given the tags of the sentence.

As $p(N^i_{j,k} \mid t_{0,n})$ can only be computed precisely after a full parse of the sentence, C&C derive several approximations, in each case starting from the well known equation for $p(N^i_{j,k} \mid t_{0,n})$ in terms of the inside and outside probabilities, $\beta(N^i_{j,k})$ and $\alpha(N^i_{j,k})$.

$$p(N^i_{j,k} \mid t_{0,n}) = \frac{\beta(N^i_{j,k})\alpha(N^i_{j,k})}{p(t_{0,n})} \qquad (1)$$

where $\beta(N^i_{j,k})$ and $\alpha(N^i_{j,k})$ are defined as follows:

$$\beta(N^i_{j,k}) = p(t_{j,k} \mid N^i_{j,k}) \qquad (2)$$

$$\alpha(N^i_{j,k}) = p(t_{0,j}, N^i_{j,k}, t_{k,n}) \qquad (3)$$

C&Cs best approximation is based upon the equation:

$$p(N^i_{j,k} \mid t_{0,n}) \approx \frac{p(N^i_{j,k} \mid t_{j-1})\beta(N^i_{j,k})p(t_k \mid N^i_{j,k})}{p(t_{j,k} \mid t_{j-1})p(t_k \mid t_{k-1})} \qquad (4)$$

Informally, this can be obtained by approximating the outside probability $\alpha(N^i_{j,k})$ in Equation 1 with a bitag estimate.

Of the five terms in Equation 4, two can be directly estimated from training data: the "boundary statistics" $p(N^i_{j,k} \mid t_j)$ (the probability of a constituent of type $N^i_{j,k}$starting just after the tag t_j) and $p(t_k \mid N^i_{j,k})$ (the probability of t_k appearing just after the end of a constituent of type $N^i_{j,k}$). The tag sequence probabilitiy in the denominator is approximated using a bi-tag approximation:

$$p(t_{j,k}) = \prod_{i=j}^{k} p(t_i \mid t_{i-1}). \qquad (5)$$

The basic algorithm then is quite simple. One uses the standard chart-parsing algorithm, except at each iteration one takes from the agenda the constituent that maximizes the FOM described in Equation 4.

There are, however, two minor complexities that need to be noted. The first relates to the inside probability $\beta(N^i_{j,k})$. C&C approximate it with the sum of the probabilities of all the parses for $N^i_{j,k}$ found at that point in the parse. This in turn requires a somewhat complicated scheme to avoid repeatedly re-evaluating Equation 4 whenever a new parse is found. In this paper we adopt a slightly simpler method. We approximate $\beta(N^i_{j,k})$ by the most probable parse for $N^i_{j,k}$, rather than the sum of all the parses. We justify this on the grounds that our parser eventually returns the most probable parse, so it seems reasonable to base our metric on its value. This also simplifies updating $\beta(N^i_{j,k})$ when new parses are found for $N^i_{j,k}$. Our algorithm compares the probability of the new parse to the best already found for $N^i_{j,k}$. If the old one is higher, nothing need be done. If the new one is higher, it is simply added to the agenda.

The second complexity has to do with the fact that in Equation 4 the probability of the tags $t_{j,k}$ are approximated using two different distributions, once in the numerator where we use the PCFG probabilities, and once in the denominator, where we use the bi-tag probabilities. One fact noted by C&C, but not discussed in their paper, is that typically the bi-tag model gives higher probabilities for a tag sequence than does the PCFG distribution. For any single tag t_j, the difference is not much, but as we use Equation 4 to compute our FOM for larger constituents, the numerator becomes smaller and smaller with respect to the denominator, effectively favoring smaller constituents. To avoid this one needs to *normalize* the two distributions to produce more similar results.

We have empirically measured the normalization factor and found that the bi-tag distribution produces probabilities that are approximately 1.3 times those produced by the PCFG distribution, on a per-word basis. We correct for this by making the PCFG probability of a known tag $\eta \geq 1$. This has the effect of multiplying the inside probability $\beta(N^i_{j,k})$ by η^{k-j}. In Section 4 we show how the behavior of our algorithm changes for ηs between 1.0 and 2.4.

3 Chart parsing and binarization

Informally, our algorithm differs from the one presented in C&C primarily in that we rank *all* edges, incomplete as well as complete, with respect to the FOM. A straight-forward way to extend C&C in this fashion is to transform the grammar so that all productions are either unary or binary. Once this has been done there is no need for incomplete edges at all in bottom-up parsing, and parsing can be performed using the CKY algorithm, suitably extended to handle unary productions.

One way to convert a PCFG into this form is *left-factoring* (Hopcroft and Ullman, 1979). Left-factoring replaces each production $A \rightarrow \beta : p$, where p is the production probability and $|\beta| = n > 2$, with the following set of binary productions:

$$\begin{array}{rcll} A & \rightarrow & `\beta_{1,n-1}' \; \beta_n : p & \\ `\beta_{1,i}' & \rightarrow & `\beta_{1,i-1}' \; \beta_i : 1.0 & \text{for } i \in [3,n] \\ `\beta_{1,2}' & \rightarrow & \beta_1 \; \beta_2 : 1.0 & \end{array}$$

In these productions β_i is the ith element of β and '$\beta_{i,j}$' is the subsequence $\beta_i \ldots \beta_j$ of β, but treated as a 'new' single non-terminal in the left-factored grammar (the quote marks indicate that this subsequence is to be considered a single symbol).

For example, the production

$$\mathsf{VP} \rightarrow \mathsf{V\ NP\ NP\ PP} : 0.7$$

left-factors to the following productions:

$$\begin{array}{rcl} \mathsf{VP} & \rightarrow & `\mathsf{V\ NP\ NP}'\ \mathsf{PP} : 0.7 \\ `\mathsf{V\ NP\ NP}' & \rightarrow & `\mathsf{V\ NP}'\ \mathsf{PP} : 1.0 \\ `\mathsf{V\ NP}' & \rightarrow & \mathsf{V\ NP} : 1.0 \end{array}$$

It is not difficult to show that the left-factored grammar defines the same probability distribution over strings as the original grammar, and to devise a tree transformation that maps each parse tree of the original grammar into a unique parse tree of the left-factored grammar of the same probability.

In fact, the assumption that all productions are at most binary is not extraordinary, since tabular parsers that construct complete parse forests in worst-case $O(n^3)$ time explicitly or implicitly convert their grammars into binary branching form (Lang, 1974; Lang, 1991).

Sikkel and Nijholt (1997) describe in detail the close relationship between the CKY algorithm, the Earley algorithm and a bottom-up

variant of the Earley algorithm. The key observation is that the 'new' non-terminals '$\beta_{1,i}$' in a CKY parse using a left-factored grammar correspond to the set of non-empty incomplete edges $A \rightarrow \beta_{1,i} \cdot \beta_{i+1,n}$ in the bottom-up variant of the Earley algorithm, where $A \rightarrow \beta_{1,n}$ is a production of the original grammar. Specifically, the fundamental rule of chart parsing (Kay, 1980), which combines an incomplete edge $A \rightarrow \alpha \cdot B\beta$ with a complete edge $B \rightarrow \gamma \cdot$ to yield the edge $A \rightarrow \alpha\, B \cdot \beta$, corresponds to the left-factored productions 'αB' $\rightarrow \alpha\, B$ if β is non-empty or $A \rightarrow$ 'α'B if β is empty. Thus in general a single 'new' non-terminal in a CKY parse using the left-factored grammar abbreviates several incomplete edges in the Earley algorithm.

4 The Experiment

For our experiment, we used a tree-bank grammar induced from sections 2–21 of the Penn Wall Street Journal text (Marcus et al., 1993), with section 22 reserved for testing. All sentences of length greater than 40 were ignored for testing purposes as done in both C&C and Goodman (1997). We applied the binarization technique described above to the grammar.

We chose to measure the amount of work done by the parser in terms of the average number of edges popped off the agenda before finding a parse. This method has the advantage of being platform independent, as well as providing a measure of "perfection". Here, perfection is the minimum number of edges we would need to pop off the agenda in order to create the correct parse. For the binarized grammar, where each popped edge is a completed constituent, this number is simply the number of terminals plus nonterminals in the sentence— on average, 47.5.

Our algorithm includes some measures to reduce the number of items on the agenda, and thus (presumably) the number of popped edges. Each time we add a constituent to the chart, we combine it with the constituents on either side of it, potentially creating several new edges. For each of these new edges, we check to see if a matching constituent (i.e. a constituent with the same head, start, and end points) already exists in either the agenda or the chart. If there is no match, we simply add the new edge to the agenda. If there is a match but the old parse

Figure 1: η vs. Popped Edges

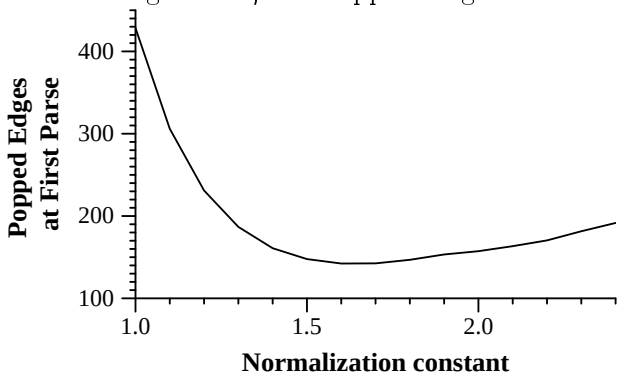
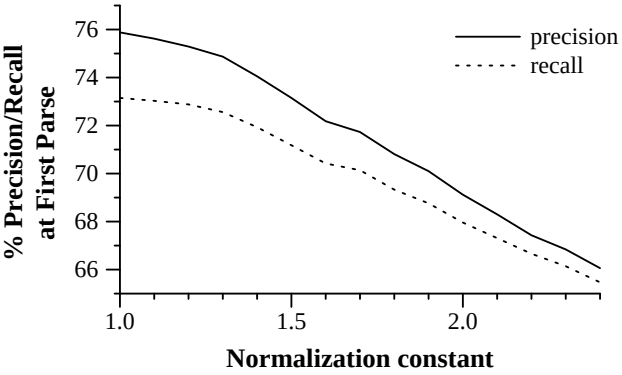


Figure 2: η vs. Precision and Recall

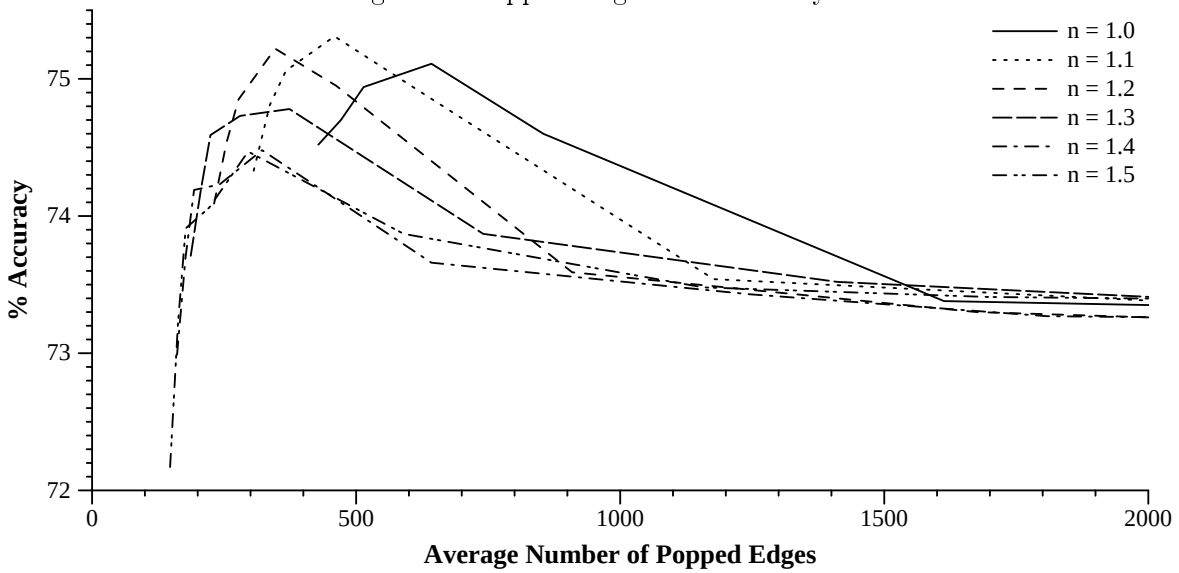


of $N^i_{j,k}$ is better than the new one, we discard the new parse. Finally, if we have found a better parse of $N^i_{j,k}$, we add the new edge to the agenda, removing the old one if it has not already been popped.

We tested the parser on section section 22 of the WSJ text with various normalization constants η, working on each sentence only until we reached the first full parse. For each sentence we recorded the number of popped edges needed to reach the first parse, and the precision and recall of that parse. The average number of popped edges to first parse as a function of η is shown in Figure 1, and the average precision and recall are shown in Figure 2.

The number of popped edges decreases as η increases from 1.0 to 1.7, then begins to increase again. See Section 5 for discussion of these results. The precision and recall also decrease as η increases. Note that, because we used a binarized grammer for parsing, the trees produced by the parser contain binarized labels rather than the labels in the treebank. In order to calculate precision and recall, we "debinarized"

Figure 3: Popped Edges vs. Accuracy



the parser's output and then calculated the figures as usual.

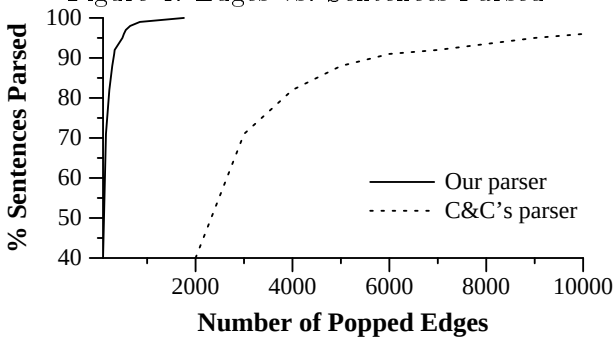
These results suggest two further questions: Is the higher accuracy with lower η due in part to the higher number of edges popped? If so, can we gain accuracy with higher η by letting the parser continue past the first parse (i.e. pop more edges)? To answer these questions, we ran the parser again, this time allowing it to continue parsing until it had popped 20 times as many edges as needed to reach the first parse. The results of this experiment are shown in Figure 3, where we plot (precision + recall)/2 (henceforth "accuracy") as a function of edges.

Note that regardless of η the accuracy of the parse increases given extra time, but that all of the increase is achieved with only 1.5 to 2 times as many edges as needed for the first parse. For η between 1.0 and 1.2, the highest accuracy is almost the same, about 75.2, but this value is reached with an average of slightly under 400 edges when $\eta = 1.2$, compared to about 650 when $\eta = 1.0$.

5 Results

To better understand the experimental results it first behooves us to compare them to those achieved previously. Goodman's results (1997) are hard to compare against ours because his parser returns more than a singe best parse and because he measures processing time, not edges. However he does give edges/second for one of his

Figure 4: Edges vs. Sentences Parsed



parsers and this plus his parsing times suggests that for him edges/sentence will measure in the tens of thousands — a far cry from our hundreds. Ratnaparki's (1997) beam search parsing procedure produces higher accuracy results than our PCFG model, and achieves this with a beam width of 20. Unfortunately his paper does not give statistics which can be directly compared with ours.

The work by C&C is easier to compare. In Figure 4 we reproduce C&C's results on the percentage of sentences (length 18-26) parsed as a function of number of edges used. We performed the same experiment, and our results are included there as well. This figure makes dramatic the order of magnitude improvement provided by our new scheme, but it is not too easy to read numbers off of it. Such numbers are provided in Table 1.

Table 1: Edges vs. Sentences Parsed

% Sents Parsed	Our Edges	C&C Edges
40	90	2000
71	150	3000
82	220	4000
91	320	6000
95	490	9000
96	520	10000
100	1760	

Our figures were obtained using $\eta = 1.2$. As can be seen, our parser requires about one twentieth the number of edges required by C&C.

Indeed, the low average number of edges to first parse is probably the most striking thing about our results. Even allowing for the fact that considerably more edges must be pushed than are popped, the total number of edges required to first parse is quite small. Since the average number of edges required to construct just the (left-factored) test corpus trees is 47.5, our parsing system considers as few as 3 times as many edges as are required to actually produce the output tree.

Almost as interesting, if η is below 1.4, the precision and recall scores of the first parse are better than those obtained by running the parser to exhaustion, even though the probability of the first parses our algorithm returns cannot be higher than that found by the exhaustive version. Furthermore, as seen in Figure 3, running our parser past the first parse by a small amount (150% of the edges required for the first parse) produces still more accurate parses. At 150% of the minimum number of edges and $\eta = 1.2$ the precision/recall figures are about 2% above those for the maximum likelihood parse.

We have two (possibly related) theories of these phenomona. It may be that the FOM metric used to select constituents forces our parser to concentrate on edges which are plausible given their surrounding preterminals; information which is ignored by the exhaustive maximum likelihood parser. Alternatively, it may be that because our FOM causes our parser to prefer edges with a high inside times (estimated) outside probability, it is in fact partially mimicking Goodman's (Goodman, 1996) 'Labelled Recall' parsing algorithm, which does not return the highest probability parse but attempts to maximize labelled bracket recall with the test set.

Finally, it is interesting to note that the minimum number of edges per parse is reached when $\eta \approx 1.65$, which is considerably larger than the theoretical estimate of 1.3 given earlier. Notice that one effect of increasing η is to raise the FOM for longer constituents. It may be that on average a partial parse is completed fastest if larger constituents receive more attention since they are more likely to lead quickly to a complete analysis, which would be one consequence of the larger than expected η.

This last hypothesis is also consistent with the observation that average precision and recall sharply falls off when η is increased beyond its theoretically optimal value, since then the parser is presumably focusing on relatively larger constituents and ignoring other, strictly more plausible, smaller ones.

6 Conclusion

It is worth noting that while we have presented the use of edge-based best-first chart parsing in the service of a rather pure form of PCFG parsing, there is no particular reason to assume that the technique is so limited in its domain of applicability. One can imagine the same techniques coupled with more informative probability distributions, such as lexicalized PCFGs (Charniak, 1997), or even grammars not based upon literal rules, but probability distributions that describe how rules are built up from smaller components (Magerman, 1995; Collins, 1997). Clearly further research is warranted.

Be this as it may, the take-home lesson from this paper is simple: combining an edge-based agenda with the figure of merit from C&C

- is easy to do by simply binarizing the grammar
- provides a factor of 20 or so reduction in the number of edges required to find a first parse, and
- improves parsing precision and recall over exhaustive parsing.

To the best of our knowledge this is currently the most effecient parsing technique for PCFG

grammars induced from large tree-banks. As such we strongly recommend this technique to others interested in PCFG parsing.

References

Robert J. Bobrow. 1990. Statistical agenda parsing. In *DARPA Speech and Language Workshop*, pages 222–224.

Sharon Caraballo and Eugene Charniak. Forthcoming. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598–603, Menlo Park. AAAI Press/MIT Press.

Mahesh V. Chitrao and Ralph Grishman. 1990. Statistical parsing of messages. In *DARPA Speech and Language Workshop*, pages 263–266.

Michael John Collins. 1997. Three generative lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 177–183.

Joshua Goodman. 1997. Global thresholding and multiple-pass parsing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 11–25.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Martin Kay. 1980. Algorithm schemata and data structures in syntactic processing. In Barbara J. Grosz, Karen Sparck Jones, and Bonnie Lynn Weber, editors, *Readings in Natural Language Processing*, pages 35–70. Morgan Kaufmann, Los Altos, California.

Fred Kochman and Joseph Kupin. 1991. Calculating the probability of a partial parse of a sentence. In *DARPA Speech and Language Workshop*, pages 273–240.

Bernard Lang. 1974. Deterministic techniques for efficient non-deterministic parsers. In *2nd Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 14, pages 225–269. Springer Verlag, Berlin.

Bernard Lang. 1991. Towards a uniform formal framework for parsing. In Masaru Tomita, editor, *Current Issues in Parsing Technology*, pages 153–172. Kluwer Academic Publishers, Dordrecht.

David M. Magerman and Mitchell P. Marcus. 1991. Parsing the voyager domain using pearl. In *DARPA Speech and Language Workshop*, pages 231–236.

David Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Scott Miller and Heidi Fox. 1994. Automatic grammar acquisition. In *Proceedings of the Human Language Technology Workshop*, pages 268–271.

Adwait Ratnaparki. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 1–10.

Klaas Sikkel and Anton Nijholt. 1997. Parsing of Context-Free languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 2: Linear Modelling: Background and Application, chapter 2, pages 61–100. Springer, Berlin.