

Multilevel Coarse-to-fine PCFG Parsing

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil,
David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths,
Jeremy Moore, Michael Pozar, and Theresa Vu

Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
ec@cs.brown.edu

Abstract

We present a PCFG parsing algorithm that uses a multilevel coarse-to-fine (mlctf) scheme to improve the efficiency of search for the best parse. Our approach requires the user to specify a sequence of nested partitions or equivalence classes of the PCFG nonterminals. We define a sequence of PCFGs corresponding to each partition, where the nonterminals of each PCFG are clusters of nonterminals of the original source PCFG. We use the results of parsing at a coarser level (i.e., grammar defined in terms of a coarser partition) to prune the next finer level. We present experiments showing that with our algorithm the work load (as measured by the total number of constituents processed) is decreased by a factor of ten with no decrease in parsing accuracy compared to standard CKY parsing with the original PCFG. We suggest that the search space over mlctf algorithms is almost totally unexplored so that future work should be able to improve significantly on these results.

1 Introduction

Reasonably accurate constituent-based parsing is fairly quick these days, if fairly quick means about a second per sentence. Unfortunately, this is still too slow for many applications. In some

cases researchers need large quantities of parsed data and do not have the hundreds of machines necessary to parse gigaword corpora in a week or two. More pressingly, in real-time applications such as speech recognition, a parser would be only a part of a much larger system, and the system builders are not keen on giving the parser one of the ten seconds available to process, say, a thirty-word sentence. Even worse, some applications require the parsing of multiple candidate strings per sentence (Johnson and Charniak, 2004) or parsing from a lattice (Hall and Johnson, 2004), and in these applications parsing efficiency is even more important.

We present here a multilevel coarse-to-fine (mlctf) PCFG parsing algorithm that reduces the complexity of the search involved in finding the best parse. It defines a sequence of increasingly more complex PCFGs, and uses the parse forest produced by one PCFG to prune the search of the next more complex PCFG. We currently use four levels of grammars in our mlctf algorithm. The simplest PCFG, which we call the *level-0* grammar, contains only one nontrivial nonterminal and is so simple that minimal time is needed to parse a sentence using it. Nonetheless, we demonstrate that it identifies the locations of correct constituents of the parse tree (the “gold constituents”) with high recall. Our level-1 grammar distinguishes only argument from modifier phrases (i.e., it has two nontrivial nonterminals), while our level-2 grammar distinguishes the four major phrasal categories (verbal, nominal, adjectival and prepositional phrases), and level 3 distinguishes all of the standard categories of the Penn treebank.

The nonterminal categories in these grammars can be regarded as clusters or equivalence classes of the original Penn treebank nonterminal categories. (In fact, we obtain these grammars by relabeling the node labels in the treebank and extracting a PCFG from this relabelled treebank in the standard fashion, but we discuss other approaches below.) We require that the partition of the nonterminals defined by the equivalence classes at level $l + 1$ be a refinement of the partition defined at level l . This means that each nonterminal category at level $l + 1$ is mapped to a unique nonterminal category at level l (although in general the mapping is many to one, i.e., each nonterminal category at level l corresponds to several nonterminal categories at level $l + 1$).

We use the correspondence between categories at different levels to prune possible constituents. A constituent is considered at level $l + 1$ only if the corresponding constituent at level l has a probability exceeding some threshold. Thus parsing a sentence proceeds as follows. We first parse the sentence with the level-0 grammar to produce a parse forest using the CKY parsing algorithm. Then for each level $l + 1$ we reparse the sentence with the level $l + 1$ grammar using the level l parse forest to prune as described above. As we demonstrate, this leads to considerable efficiency improvements.

The paper proceeds as follows. We next discuss previous work (Section 2). Section 3 outlines the algorithm in more detail. Section 4 presents some experiments showing that the work load (as measured by the total number of constituents processed) is decreased by a factor of ten over standard CKY parsing at the final level. We also discuss some fine points of the results therein. Finally in section 5 we suggest that because the search space of mlctf algorithms is, at this point, almost totally unexplored, future work should be able to improve significantly on these results.

2 Previous Research

Coarse-to-fine search is an idea that has appeared several times in the literature of computational linguistics and related areas. The

first appearance of this idea we are aware of is in Maxwell and Kaplan (1993), where a covering CFG is automatically extracted from a more detailed unification grammar and used to identify the possible locations of constituents in the more detailed parses of the sentence. Maxwell and Kaplan use their covering CFG to prune the search of their unification grammar parser in essentially the same manner as we do here, and demonstrate significant performance improvements by using their coarse-to-fine approach.

The basic theory of coarse-to-fine approximations and dynamic programming in a stochastic framework is laid out in Geman and Kochanek (2001). This paper describes the multilevel dynamic programming algorithm needed for coarse-to-fine analysis (which they apply to decoding rather than parsing), and show how to perform *exact coarse-to-fine* computation, rather than the heuristic search we perform here.

A paper closely related to ours is Goodman (1997). In our terminology, Goodman’s parser is a two-stage ctf parser. The second stage is a standard tree-bank parser while the first stage is a regular-expression approximation of the grammar. Again, the second stage is constrained by the parses found in the first stage. Neither stage is smoothed. The parser of Charniak (2000) is also a two-stage ctf model, where the first stage is a smoothed Markov grammar (it uses up to three previous constituents as context), and the second stage is a lexicalized Markov grammar with extra annotations about parents and grandparents. The second stage explores all of the constituents not pruned out after the first stage. Related approaches are used in Hall (2004) and Charniak and Johnson (2005).

A quite different approach to parsing efficiency is taken in Caraballo and Charniak (1998) (and refined in Charniak et al. (1998)). Here efficiency is gained by using a standard chart-parsing algorithm and pulling constituents off the agenda according to (an estimate of) their probability given the sentence. This probability is computed by estimating Equation 1:

$$p(n_{i,j}^k | s) = \frac{\alpha(n_{i,j}^k)\beta(n_{i,j}^k)}{p(s)}. \quad (1)$$

It must be estimated because during the bottom-up chart-parsing algorithm, the true outside probability cannot be computed. The results cited in Caraballo and Charniak (1998) cannot be compared directly to ours, but are roughly in the same equivalence class. Those presented in Charniak et al. (1998) are superior, but in Section 5 below we suggest that a combination of the techniques could yield better results still.

Klein and Manning (2003a) describe efficient A* for the most likely parse, where pruning is accomplished by using Equation 1 and a true upper bound on the outside probability. While their maximum is a looser estimate of the outside probability, it is an admissible heuristic and together with an A* search is guaranteed to find the best parse first. One question is if the guarantee is worth the extra search required by the looser estimate of the true outside probability.

Tsuruoka and Tsujii (2004) explore the framework developed in Klein and Manning (2003a), and seek ways to minimize the time required by the heap manipulations necessary in this scheme. They describe an iterative deepening algorithm that does not require a heap. They also speed computation by precomputing more accurate upper bounds on the outside probabilities of various kinds of constituents. They are able to reduce by half the number of constituents required to find the best parse (compared to CKY).

Most recently, McDonald et al. (2005) have implemented a dependency parser with good accuracy (it is almost as good at dependency parsing as Charniak (2000)) and very impressive speed (it is about ten times faster than Collins (1997) and four times faster than Charniak (2000)). It achieves its speed in part because it uses the Eisner and Satta (1999) algorithm for n^3 bilinear parsing, but also because dependency parsing has a much lower grammar constant than does standard PCFG parsing — after all, there are no phrasal constituents to consider. The current paper can be thought of as a way to take the sting out of the grammar constant for PCFGs by parsing first with very few phrasal constituents and adding them only

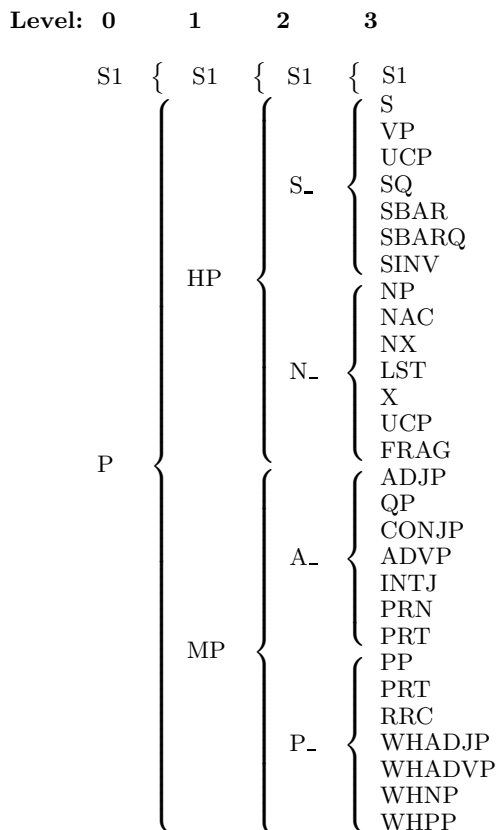


Figure 1: The levels of nonterminal labels

after most constituents have been pruned away.

3 Multilevel Course-to-fine Parsing

We use as the underlying parsing algorithm a reasonably standard CKY parser, modified to allow unary branching rules.

The complete nonterminal clustering is given in Figure 1. We do not cluster preterminals. These remain fixed at all levels to the standard Penn-tree-bank set Marcus et al. (1993).

Level-0 makes two distinctions, the root node and everybody else. At level 1 we make one further distinction, between phrases that tend to be heads of constituents (NPs, VPs, and Ss) and those that tend to be modifiers (ADJPs, PPs, etc.). Level-2 has a total of five categories: root, things that are typically headed by nouns, those headed by verbs, things headed by prepositions, and things headed by classical modifiers (adjectives, adverbs, etc.). Finally, level 3 is the

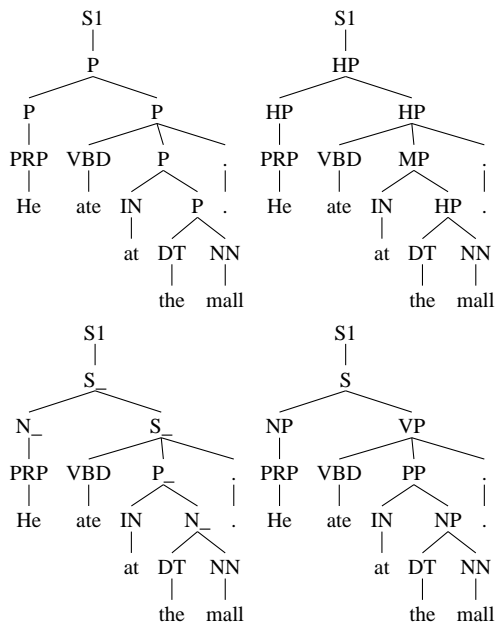


Figure 2: A tree represented at levels 0 to 3

classical tree-bank set. As an example, Figure 2 shows the parse for the sentence “He ate at the mall.” at levels 0 to 3.

During training we create four grammars, one for each level of granularity. So, for example, at level 1 the tree-bank rule

$$S \rightarrow NP VP .$$

would be translated into the rule

$$HP \rightarrow HP HP .$$

That is, each constituent type found in “ $S \rightarrow NP VP .$ ” is mapped into its generalization at level 1. The probabilities of all rules are computed using maximum likelihood for constituents at that level.

The grammar used by the parser can best be described as being influenced by four components:

1. the nonterminals defined at that level of parsing,
2. the binarization scheme,
3. the generalizations defined over the binarization, and

4. extra annotation to improve parsing accuracy.

The first of these has already been covered. We discuss the other three in turn.

In anticipation of eventually lexicalizing the grammar we binarize from the head out. For example, consider the rule

$$A \rightarrow a b c d e$$

where c is the head constituent. We binarize this as follows:

$$\begin{aligned} A &\rightarrow A_1 e \\ A_1 &\rightarrow A_2 d \\ A_2 &\rightarrow a A_3 \\ A_3 &\rightarrow b c \end{aligned}$$

Grammars induced in this way tend to be too specific, as the binarization introduce a very large number of very specialized phrasal categories (the A_i). Following common practice Johnson (1998; Klein and Manning (2003b) we Markovize by replacing these nonterminals with ones that remember less of the immediate rule context. In our version we keep track of only the parent, the head constituent and the constituent immediately to the right or left, depending on which side of the constituent we are processing. With this scheme the above rules now look like this:

$$\begin{aligned} A &\rightarrow A_{d,c} e \\ A_{d,c} &\rightarrow A_{a,c} d \\ A_{a,c} &\rightarrow a A_{b,c} \\ A_{b,c} &\rightarrow b c \end{aligned}$$

So, for example, the rule “ $A \rightarrow A_{d,c} e$ ” would have a high probability if constituents of type A , with c as their head, often have d followed by e at their end.

Lastly, we add parent annotation to phrasal categories to improve parsing accuracy. If we assume that in this case we are expanding a rule for an A used as a child of Q , and a, b, c, d , and e are all phrasal categories, then the above rules become:

$$\begin{aligned} A^Q &\rightarrow A_{d,c} e^A \\ A_{d,c} &\rightarrow A_{a,c} d^A \\ A_{a,c} &\rightarrow a^A A_{b,c} \\ A_{b,c} &\rightarrow b^A c^A \end{aligned}$$

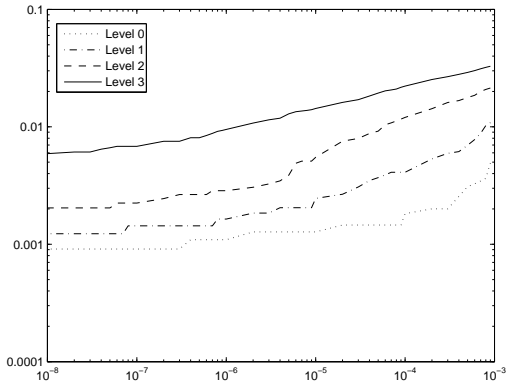


Figure 3: Probability of a gold constituent being pruned as a function of pruning thresholds for the first 100 sentences of the development corpus

Once we have parsed at a level, we evaluate the probability of a constituent $p(n_{i,j}^k | s)$ according to the standard inside-outside formula of Equation 1. In this equation $n_{i,j}^k$ is a constituent of type k spanning the words i to j , and $\alpha(\cdot)$ and $\beta(\cdot)$ are the outside and inside probabilities of the constituent, respectively. Because we prune at the end each granularity level, we can evaluate the equation exactly; no approximations are needed (as in, e.g., Charniak et al. (1998)).

During parsing, instead of building each constituent allowed by the grammar, we first test if the probability of the corresponding coarser constituent (which we have from Equation 1 in the previous round of parsing) is greater than a threshold. (The threshold is set empirically based upon the development data.) If it is below the threshold, we do not put the constituent in the chart. For example, before we can use a NP and a VP to create a S (using the rule above), we would first need to check that the probability in the coarser grammar of using the same span HP and HP to create a HP is above the threshold. We use the standard inside-outside formula to calculate this probability (Equation 1). The empirical results below justify our conjecture that there are thresholds that allow significant pruning while leaving the gold constituents untouched.

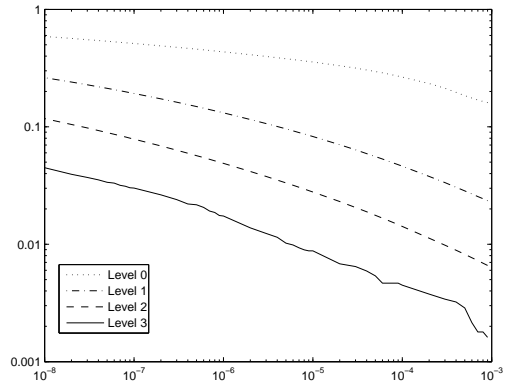


Figure 4: Fraction of incorrect constituents kept as a function of pruning thresholds for the first 100 sentences of the development corpus

4 Results

In all experiments the system is trained on the Penn tree-bank sections 2-21. Section 23 is used for testing and section 24 for development. The input to the parser are the gold-standard parts of speech, not the words.

The point of parsing at multiple levels of granularity is to prune the results of rough levels before going on to finer levels. In particular, it is necessary for any pruning scheme to retain the true (gold-standard WSJ) constituents in the face of the pruning. To gain an idea of what is possible, consider Figure 3. According to the graph, at the zeroth level of parsing and a the pruning level 10^{-4} the probability that a gold constituent is deleted due to pruning is slightly more than 0.001 (or 0.1%). At level three it is slightly more that 0.01 (or 1.0%).

The companion figure, Figure 4 shows the retention rate of the non-gold (incorrect) constituents. Again, at pruning level 10^{-4} and parsing level 0 we retain about .3 (30%) of the bad constituents (so we pruned 70%), whereas at level 3 we retain about .004 (0.4%). Note that in the current paper we do not actually prune at level 3, instead return the Viterbi parse. We include pruning results here in anticipation of future work in which level 3 would be a precursor to still more fine-grained parsing.

As noted in Section 2, there is some (implicit)

Level	Constits Produced	Constits Pruned	% Pruned
	$\cdot 10^6$	$\cdot 10^6$	
0	8.82	7.55	86.5
1	9.18	6.51	70.8
2	11.2	9.48	84.4
3	11.8	0	0.0
total	40.4	–	–
3-only	392.0	0	0

Figure 5: Total constituents pruned at all levels for WSJ section 23, sentences of length ≤ 100

debate in the literature on using estimates of the outside probability in Equation 1, or instead computing the exact upper bound. The idea is that an exact upper bound gives one an admissible search heuristic but at a cost, since it is a less accurate estimator of the true outside probability. (Note that even the upper bound does not, in general, keep *all* of the gold constituents, since a non-perfect model will assign some of them low probability.) As is clear from Figure 3, the estimate works very well indeed.

On the basis of this graph, we set the lowest allowable constituent probability at $\geq 5 \cdot 10^{-4}$, $\geq 10^{-5}$, and $\geq 10^{-4}$ for levels 0, 1, and 2, respectively. No pruning is done at level 3, since there is no level 4. After setting the pruning parameters on the development set we proceed to parse the test set (WSJ section 23). Figure 5 shows the resulting pruning statistics. The total number of constituents created at level 0, for all sentences combined, is $8.82 \cdot 10^6$. Of those $7.55 \cdot 10^6$ (or 86.5%) are pruned before going on to level 1. At level 1, the 1.3 million left over from level 0 expanded to a total of $9.18 \cdot 10^6$. 70.8% of these in turn are pruned, and so forth. The percent pruned at, e.g., level 1 in Figure 3 is much higher than that shown here because it considers *all* of the possible level-1 constituents, not just those left unpruned after level 0.

There is no pruning at level 3. There we simply return the Viterbi parse. We also show that with pruning we generate a total of $40.4 \cdot 10^6$ constituents. For comparison exhaustively parsing using the tree-bank grammar yields a total of $392 \cdot 10^6$ constituents. This is the factor-of-10

Level	Time for Level	Running Total
0	1598	1598
1	2570	4168
2	4303	8471
3	1527	9998
3-only	114654	–

Figure 6: Running times in seconds on WSJ section 23, with and without pruning

workload reduction mentioned in Section 1.

There are two points of interest. The first is that each level of pruning is worthwhile. We do not get most of the effect from one or the other level. The second point is that we get significant pruning at level 0. The reader may remember that level 0 distinguishes only between the root node and the rest. We initially expected that it would be too coarse to distinguish good from bad constituents at this level, but it proved as useful as the other levels. The explanation is that this level does use the full tree-bank preterminal tags, and in many cases these alone are sufficient to make certain constituents *very* unlikely. For example, what is the probability of *any* constituent of length two or greater ending in a preposition? The answer is: very low. Similarly for constituents of length two or greater ending in modal verbs, and determiners. Not quite so improbable, but nevertheless less likely than most, would be constituents ending in verbs, or ending just short of the end of the sentence.

Figure 6 shows how much time is spent at each level of the algorithm, along with a running total of the time spent to that point. (This is for all sentences in the test set, length ≤ 100 .) The number for the unpruned parser is again about ten times that for the pruned version, but the number for the standard CKY version is probably too high. Because our CKY implementation is quite slow, we ran the unpruned version on many machines and summed the results. In all likelihood at least some of these machines were overloaded, a fact that our local job distributor would not notice. We suspect that the real number is significantly lower, though still

No pruning	77.9
With pruning	77.9
Klein and Manning (2003b)	77.4

Figure 7: Labeled precision/recall f-measure, WSJ section 23, all sentences of length ≤ 100

much higher than the pruned version.

Finally Figure 7 shows that our pruning is accomplished without loss of accuracy. The results with pruning include four sentences that did not receive any parses at all. These sentences received zeros for both precision and recall and presumably lowered the results somewhat. We allowed ourselves to look at the first of these, which turned out to contain the phrase:

(NP ... (INTJ (UH oh) (UH yes)) ...)

The training data does not include interjections consisting of two “UH”s, and thus a gold parse cannot be constructed. Note that a different binarization scheme (e.g. the one used in Klein and Manning (2003b)) would have smoothed over this problem. In our case the unpruned version is able to patch together a lot of *very* unlikely constituents to produce a parse, but not a very good one. Thus we attribute the problem not to pruning, but to binarization.

We also show the results for the most similar Klein and Manning (2003b) experiment. Our results are slightly better. We attribute the difference to the fact that we have the gold tags and they do not, but their binarization scheme does not run into the problems that we encountered.

5 Conclusion and Future Research

We have presented a novel parsing algorithm based upon the coarse-to-fine processing model. Several aspects of the method recommend it. First, unlike methods that depend on best-first search, the method is “holistic” in its evaluation of constituents. For example, consider the impact of parent labeling. It has been repeatedly shown to improve parsing accuracy (Johnson, 1998; Charniak, 2000; Klein and Manning, 2003b), but it is difficult if not impossible to

integrate with best-first search in bottom-up chart-parsing (as in Charniak et al. (1998)). The reason is that when working bottom up it is difficult to determine if, say, s^{sbar} is any more or less likely than s^s , as the evidence, working bottom up, is negligible. Since our method computes the exact outside probability of constituents (albeit at a coarser level) all of the top down information is available to the system. Or again, another very useful feature in English parsing is the knowledge that a constituent ends at the right boundary (minus punctuation) of a string. This can be included only in an ad-hoc way when working bottom up, but could be easily added here.

Many aspects of the current implementation that are far from optimal. It seems clear to us that extracting the maximum benefit from our pruning would involve taking the unpruned constituents and specifying them in all possible ways allowed by the next level of granularity. What we actually did is to propose all possible constituents at the next level, and immediately rule out those lacking a corresponding constituent remaining at the previous level. This was dictated by ease of implementation. Before using mlctf parsing in a production parser, the other method should be evaluated to see if our intuitions of greater efficiency are correct.

It is also possible to combine mlctf parsing with queue reordering methods. The best-first search method of Charniak et al. (1998) estimates Equation 1. Working bottom up, estimating the inside probability is easy (we just sum the probability of all the trees found to build this constituent). All the cleverness goes into estimating the outside probability. Quite clearly the current method could be used to provide a more accurate estimate of the outside probability, namely the outside probability at the coarser level of granularity.

There is one more future-research topic to add before we stop, possibly the most interesting of all. The particular tree of coarser to finer constituents that governs our mlctf algorithm (Figure 1) was created by hand after about 15 minutes of reflection and survives, except for typos, with only two modifications. There is no rea-

son to think it is anywhere close to optimal. It should be possible to define “optimal” formally and search for the best mlctf constituent tree. This would be a clustering problem, and, fortunately, one thing statistical NLP researchers know how to do is cluster.

Acknowledgments

This paper is the class project for Computer Science 241 at Brown University in fall 2005. The faculty involved were supported in part by DARPA GALE contract HR0011-06-2-0001. The graduate students were mostly supported by Brown University fellowships. The undergraduates were mostly supported by their parents. Our thanks to all.

References

- Sharon Caraballo and Eugene Charniak. 1998. Figures of merit for best-first probabilistic parsing. *Computational Linguistics*, 24(2):275–298.
- Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 2005 Meeting of the Association for Computational Linguistics*.
- Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 127–133. Morgan Kaufmann.
- Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.
- Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, San Francisco. Morgan Kaufmann.
- Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–464.
- Stuart Geman and Kevin Kochanek. 2001. Dynamic programming and the representation of soft-decodable codes. *IEEE Transactions on Information Theory*, 47:549–568.
- Joshua Goodman. 1997. Global thresholding and multiple-pass parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 1997)*.
- Keith Hall and Mark Johnson. 2004. Attention shifting for parsing speech. In *The Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics*, pages 40–46.
- Keith Hall. 2004. *Best-first Word-lattice Parsing: Techniques for Integrated Syntactic Language Modeling*. Ph.D. thesis, Brown University.
- Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy-channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 33–39.
- Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.
- Dan Klein and Chris Manning. 2003a. A* parsing: Fast exact viterbi parse selection. In *Proceedings of HLT-NAACL ’03*.
- Dan Klein and Christopher Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.
- Michell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- John T. Maxwell and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590.
- Ryan McDonald, Toby Crammer, and Fernando Pereira. 2005. Online large margin training of dependency parsers. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*.
- Yoshimasa Tsuruoka and Jun’ichi Tsujii. 2004. Iterative cky parsing for probabilistic context-free grammars. In *International Joint Conference on Natural-Language Processing*.