Effective Self-Training for Parsing

David McClosky, Eugene Charniak, and Mark Johnson
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{dmcc|ec|mj}@cs.brown.edu

Abstract

We present a simple, but surprisingly effective, method of self-training a two-phase parser-reranker system using readily available unlabeled data. We show that this type of bootstrapping is possible for parsing when the bootstrapped parses are processed by a discriminative reranker. Our improved model achieves an f-score of 92.1%, an absolute 1.1% improvement (12% error reduction) over the previous best result for Wall Street Journal parsing. Finally, we provide some analysis to better understand the phenomenon.

1 Introduction

In parsing, we attempt to uncover the syntactic structure from a string of words. Much of the challenge of this lies in extracting the appropriate parsing decisions from textual examples. Given sufficient labelled data, there are several "supervised" techniques of training high-performance parsers (Charniak and Johnson, 2005; Collins, 2000; Henderson, 2004). Other methods are "semi-supervised" where they use some labelled data to annotate unlabeled data. Examples of this include self-training (Charniak, 1997) and co-training (Blum and Mitchell, 1998; Steedman et al., 2003). Finally, there are "unsupervised" strategies where no data is labeled and all annotations (including the grammar itself) must be discovered (Klein and Manning, 2002).

Semi-supervised and unsupervised methods are important because good labeled data is expensive, whereas there is no shortage of unlabeled data. While some domain-language pairs have quite a bit of labelled data (e.g. news text in English), many other categories are not as fortunate. Less unsupervised methods are more likely to be portable to these new domains, since they do not rely as much on existing annotations.

2 Previous work

A simple method of incorporating unlabeled data into a new model is *self-training*. In self-training, the existing model first labels unlabeled data. The newly labeled data is then treated as truth and combined with the actual labeled data to train a new model. This process can be iterated over different sets of unlabeled data if desired. It is not surprising that self-training is not normally effective: Charniak (1997) and Steedman et al. (2003) report either minor improvements or significant damage from using self-training for parsing. Clark et al. (2003) applies self-training to POS-tagging and reports the same outcomes. One would assume that errors in the original model would be amplified in the new model.

Parser adaptation can be framed as a semi-supervised or unsupervised learning problem. In parser adaptation, one is given annotated training data from a source domain and unannotated data from a target. In some cases, some annotated data from the target domain is available as well. The goal is to use the various data sets to produce a model that accurately parses the target domain data despite seeing little or no annotated data from that domain. Gildea (2001) and Bacchiani et al. (2006) show that out-of-domain training data can improve parsing ac-

curacy. The unsupervised adaptation experiment by Bacchiani et al. (2006) is the only successful instance of parsing self-training that we have found. Our work differs in that all our data is in-domain while Bacchiani et al. uses the Brown corpus as labelled data. These correspond to different scenarios. Additionally, we explore the use of a reranker.

Co-training is another way to train models from unlabeled data (Blum and Mitchell, 1998). Unlike self-training, co-training requires multiple learners, each with a different "view" of the data. When one learner is confident of its predictions about the data, we apply the predicted label of the data to the training set of the other learners. A variation suggested by Dasgupta et al. (2001) is to add data to the training set when multiple learners agree on the label. If this is the case, we can be more confident that the data was labelled correctly than if only one learner had labelled it.

Sarkar (2001) and Steedman et al. (2003) investigated using co-training for parsing. These studies suggest that this type of co-training is most effective when small amounts of labelled training data is available. Additionally, co-training for parsing can be helpful for parser adaptation.

3 Experimental Setup

Our parsing model consists of two phases. First, we use a generative parser to produce a list of the top n parses. Next, a discriminative reranker reorders the n-best list. These components constitute two views of the data, though the reranker's view is restricted to the parses suggested by the first-stage parser. The reranker is not able to suggest new parses and, moreover, uses the probability of each parse tree according to the parser as a feature to perform the reranking. Nevertheless, the reranker's value comes from its ability to make use of more powerful features.

3.1 The first-stage 50-best parser

The first stage of our parser is the lexicalized probabilistic context-free parser described in (Charniak, 2000) and (Charniak and Johnson, 2005). The parser's grammar is a smoothed third-order Markov grammar, enhanced with lexical heads, their parts of speech, and parent and grandparent information. The parser uses five probability distributions,

one each for heads, their parts-of-speech, head-constituent, left-of-head constituents, and right-of-head constituents. As all distributions are conditioned with five or more features, they are all heavily backed off using Chen back-off (the *average-count* method from Chen and Goodman (1996)). Also, the statistics are lightly pruned to remove those that are statistically less reliable/useful. As in (Charniak and Johnson, 2005) the parser has been modified to produce n-best parses. However, the n-best parsing algorithm described in that paper has been replaced by the much more efficient algorithm described in (Jimenez and Marzal, 2000; Huang and Chang, 2005).

3.2 The MaxEnt Reranker

The second stage of our parser is a Maximum Entropy reranker, as described in (Charniak and Johnson, 2005). The reranker takes the 50-best parses for each sentence produced by the first-stage 50-best parser and selects the best parse from those 50 parses. It does this using the reranking methodology described in Collins (2000), using a Maximum Entropy model with Gaussian regularization as described in Johnson et al. (1999). Our reranker classifies each parse with respect to 1,333,519 features (most of which only occur on few parses). The features consist of those described in (Charniak and Johnson, 2005), together with an additional 601,577 features. These features consist of the parts-of-speech, possibly together with the words, that surround (i.e., precede or follow) the left and right edges of each constituent. The features actually used in the parser consist of all singletons and pairs of such features that have different values for at least one of the best and non-best parses of at least 5 sentences in the training data. There are 147,456 such features involving only parts-of-speech and 454,101 features involving parts-of-speech and words. These additional features are largely responsible for improving the reranker's performance on section 23 to 91.3% f-score (Charniak and Johnson (2005) reported an f-score of 91.0% on section 23).

3.3 Corpora

Our labeled data comes from the Penn Treebank (Marcus et al., 1993) and consists of about 40,000 sentences from Wall Street Journal (WSJ) articles

annotated with syntactic information. We use the standard divisions: Sections 2 through 21 are used for training, section 24 is held-out development, and section 23 is used for final testing. Our unlabeled data is the North American News Text corpus, NANC (Graff, 1995), which is approximately 24 million unlabeled sentences from various news sources. NANC contains no syntactic information. Sentence boundaries in NANC are induced by a simple discriminative model. We also perform some basic cleanups on NANC to ease parsing. NANC contains news articles from various news sources including the Wall Street Journal, though for this paper, we only use articles from the LA Times.

4 Experimental Results

We use the reranking parser to produce 50-best parses of unlabeled news articles from NANC. Next, we produce two sets of one-best lists from these 50-best lists. The parser-best and reranker-best lists represent the best parse for each sentence according to the parser and reranker, respectively. Finally, we mix a portion of parser-best or reranker-best lists with the standard Wall Street Journal training data (sections 2-21) to retrain a new parser (but not reranker[1]) model. The Wall Street Journal training data is combined with the NANC data in the following way: The count of each parsing event is the (optionally weighted) sum of the counts of that event in Wall Street Journal and NANC. Bacchiani et al. (2006) show that count merging is more effective than creating multiple models and calculating weights for each model (model interpolation). Intuitively, this corresponds to concatenating our training sets, possibly with multiple copies of each to account for weighting.

Some notes regarding evaluations: All numbers reported are f-scores[2]. In some cases, we evaluate only the parser's performance to isolate it from the reranker. In other cases, we evaluate the reranking parser as a whole. In these cases, we will use the term *reranking parser*.

Table 1 shows the difference in parser's (not reranker's) performance when trained on parser-best

---

[1]We attempted to retrain the reranker using the self-trained sentences, but found no significant improvement.

[2]The harmonic mean of labeled precision (P) and labeled recall (R), i.e. $f = \frac{2 \times P \times R}{P+R}$

Sentences added	Parser-best	Reranker-best
0 (baseline)	90.3	
50k	90.1	90.7
250k	90.1	90.7
500k	90.0	90.9
750k	89.9	91.0
1,000k	90.0	90.8
1,500k	90.0	90.8
2,000k	–	91.0

Table 1: f-scores after adding either parser-best or reranker-best sentences from NANC to WSJ training data. While the reranker was used to produce the reranker-best sentences, we performed this evaluation using only the first-stage parser to parse all sentences from section 22. We did not train a model where we added 2,000k parser-best sentences.

output versus reranker-best output. Adding parser-best sentences recreates previous self-training experiments and confirms that it is not beneficial. However, we see a large improvement from adding reranker-best sentences. One may expect to see a monotonic improvement from this technique, but this is not quite the case, as seen when we add 1,000k sentences. This may be due to some sections of NANC being less similar to WSJ or containing more noise. Another possibility is that these sections contains harder sentences which we cannot parse as accurately and thus are not as useful for self-training. For our remaining experiments, we will only use reranker-best lists.

We also attempt to discover the optimal number of sentences to add from NANC. Much of the improvement comes from the addition of the initial 50,000 sentences, showing that even small amounts of new data can have a significant effect. As we add more data, it becomes clear that the maximum benefit to parsing accuracy by strictly adding reranker-best sentences is about 0.7% and that f-scores will asymptote around 91.0%. We will return to this when we consider the relative weightings of WSJ and NANC data.

One hypothesis we consider is that the reranked NANC data incorporated some of the features from the reranker. If this were the case, we would not see an improvement when evaluating a reranking parser

Sentences added	1	22	24
0 (baseline)	91.8	92.1	90.5
50k	91.8	92.4	90.8
250k	91.8	92.3	91.0
500k	92.0	92.4	90.9
750k	92.0	92.4	91.1
1,000k	92.1	92.2	91.3
1,500k	92.1	92.1	91.2
1,750k	92.1	92.0	91.3
2,000k	92.2	92.0	91.3

Table 2: f-scores from evaluating the reranking parser on three held-out sections after adding reranked sentences from NANC to WSJ training. These evaluations were performed on all sentences.

on the same models. In Table 2, we see that the new NANC data contains some information orthogonal to the reranker and improves parsing accuracy of the reranking parser.

Up to this point, we have only considered giving our true training data a relative weight of one. Increasing the weight of the Wall Street Journal data should improve, or at least not hurt, parsing performance. Indeed, this is the case for both the parser (figure not shown) and reranking parser (Figure 1). Adding more weight to the Wall Street Journal data ensures that the counts of our events will be closer to our more accurate data source while still incorporating new data from NANC. While it appears that the performance still levels off after adding about one million sentences from NANC, the curves corresponding to higher WSJ weights achieve a higher asymptote. Looking at the performance of various weights across sections 1, 22, and 24, we decided that the best combination of training data is to give WSJ a relative weight of 5 and use the first 1,750k reranker-best sentences from NANC.

Finally, we evaluate our new model on the test section of Wall Street Journal. In Table 3, we note that baseline system (i.e. the parser and reranker trained purely on Wall Street Journal) has improved by 0.3% over Charniak and Johnson (2005). The 92.1% f-score is the 1.1% absolute improvement mentioned in the abstract. The improvement from self-training is significant in both macro and micro tests ($p < 10^{-5}$).
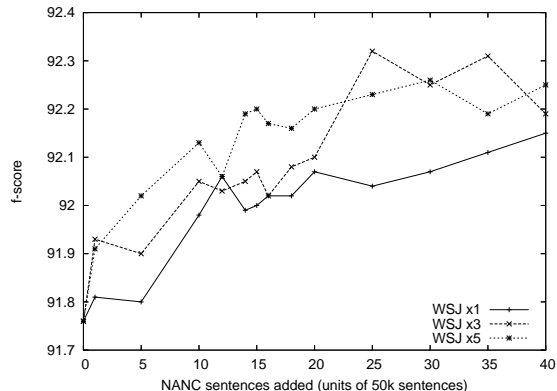

Figure 1: Effect of giving more relative weight to WSJ training data on reranking parser f-score. Evaluations were done from all sentences from section 1.

Model	f_{parser}	$f_{reranker}$
Charniak and Johnson (2005)	–	91.0
Current baseline	89.7	91.3
WSJ + NANC	91.0	92.1

Table 3: f-scores on WSJ section 23. f_{parser} and $f_{reranker}$ are the evaluation of the parser and reranking parser on all sentences, respectively. "WSJ + NANC" represents the system trained on WSJ training (with a relative weight of 5) and 1,750k sentences from the reranker-best list of NANC.

5 Analysis

We performed several types of analysis to better understand why the new model performs better. We first look at global changes, and then at changes at the sentence level.

5.1 Global Changes

It is important to keep in mind that while the reranker seems to be key to our performance improvement, the reranker per se never sees the extra data. It only sees the 50-best lists produced by the first-stage parser. Thus, the nature of the changes to this output is important.

We have already noted that the first-stage parser's one-best has significantly improved (see Table 1). In Table 4, we see that the 50-best oracle rate also im-

Model	1-best	10-best	50-best
Baseline	89.0	94.0	95.9
WSJ×1 + 250k	89.8	94.6	96.2
WSJ×5 + 1,750k	90.4	94.8	96.4

Table 4: Oracle f-scores of top n parses produced by baseline, a small self-trained parser, and the "best" parser

proves from 95.5% for the original first-stage parser, to 96.4% for our final model. We do not show it here, but if we self-train using first-stage one-best, there is no change in oracle rate.

The first-stage parser also becomes more "decisive." The average (geometric mean) of $\log_2(\text{Pr}(1\text{-best}) / \text{Pr}(50\text{th-best}))$ (i.e. the ratios between the probabilities in log space) increases from 11.959 for the baseline parser, to 14.104 for the final parser. We have seen earlier that this "confidence" is deserved, as the first-stage one-best is so much better.

5.2 Sentence-level Analysis

To this point we have looked at bulk properties of the data fed to the reranker. It has higher one best and 50-best-oracle rates, and the probabilities are more skewed (the higher probabilities get higher, the lows get lower). We now look at sentence-level properties. In particular, we analyzed the parsers' behavior on 5,039 sentences in sections 1, 22 and 24 of the Penn treebank. Specifically, we classified each sentence into one of three classes: those where the self-trained parser's f-score increased relative to the baseline parser's f-score, those where the f-score remained the same, and those where the self-trained parser's f-score decreased relative to the baseline parser's f-score. We analyzed the distribution of sentences into these classes with respect to four factors: sentence length, the number of unknown words (i.e., words not appearing in sections 2–21 of the Penn treebank) in the sentence, the number of coordinating conjunctions (CC) in the sentence, and the number of prepositions (IN) in the sentence. The distributions of classes (better, worse, no change) with respect to each of these factors individually are graphed in Figures 2 to 5.

Figure 2 shows how the self-training affects f-score as a function of sentence length. The top line
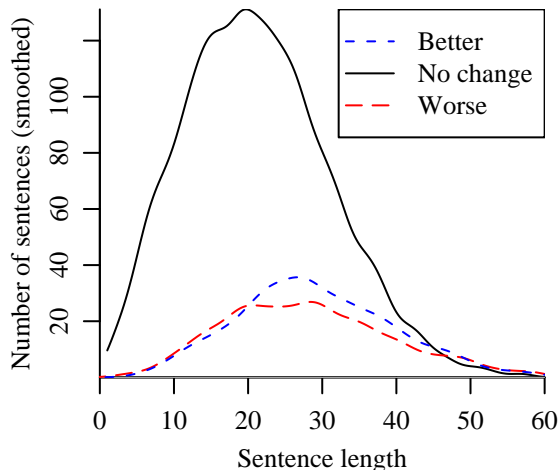

Figure 2: How self-training improves performance as a function of sentence length

shows that the f-score of most sentences remain unchanged. The middle line is the number of sentences that improved their f-score, and the bottom are those which got worse. So, for example, for sentences of length 30, about 80 were unchanged, 25 improved, and 22 worsened. It seems clear that there is no improvement for either very short sentences, or for very long ones. (For long ones the graph is hard to read. We show a regression analysis later in this section that confirms this statement.) While we did not predict this effect, in retrospect it seems reasonable. The parser was already doing very well on short sentences. The very long ones are hopeless, and the middle ones are just right. We call this the Goldilocks effect.

As for the other three of these graphs, their stories are by no means clear. Figure 3 seems to indicate that the number of unknown words in the sentence does *not* predict that the reranker will help. Figure 4 might indicate that the self-training parser improves prepositional-phrase attachment, but the graph looks suspiciously like that for sentence length, so the improvements might just be due to the Goldilocks effect. Finally, the improvement in Figure 5 is hard to judge.

To get a better handle on these effects we did a factor analysis. The factors we consider are number of CCs, INs, and unknowns, plus sentence length. As Figure 2 makes clear, the relative performance of the self-trained and baseline parsers does not
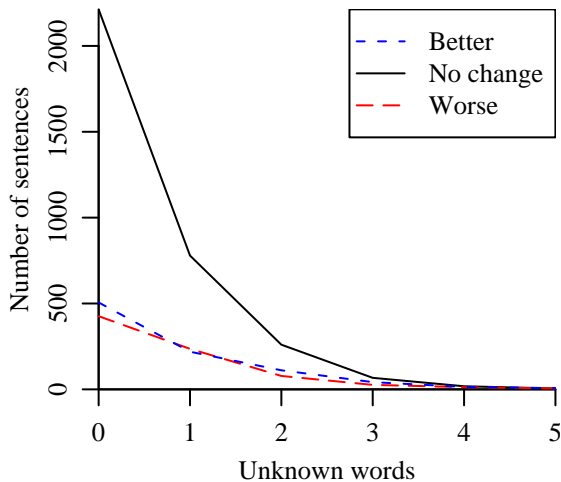
Figure 3: How self-training improves performance as a function of number of unknown words

	Estimate	Pr(> 0)
(Intercept)	-0.25328	0.3649
BinnedLength(10,20]	0.02901	0.9228
BinnedLength(20,30]	0.45556	0.1201
BinnedLength(30,40]	0.40206	0.1808
BinnedLength(40,50]	0.26585	0.4084
BinnedLength(50,200]	-0.06507	0.8671
CCs	0.12333	0.0541

Table 5: Factor analysis for the question: does the self-trained parser improve the parse with the highest probability

vary linearly with sentence length, so we introduced binned sentence length (with each bin of length 10) as a factor.

Because the self-trained and baseline parsers produced equivalent output on 3,346 (66%) of the sentences, we restricted attention to the 1,693 sentences on which the self-trained and baseline parsers' f-scores differ. We asked the program to consider the following factors: binned sentence length, number of PPs, number of unknown words, and number of CCs. The results are shown in Table 5. The factor analysis is trying to model the log odds as a sum of linearly weighted factors. I.e,

$$log(P(1|x)/(1 - P(1|x))) \quad = \quad \alpha_0 + \sum_{j=1}^{m} \alpha_j f_j(x)$$

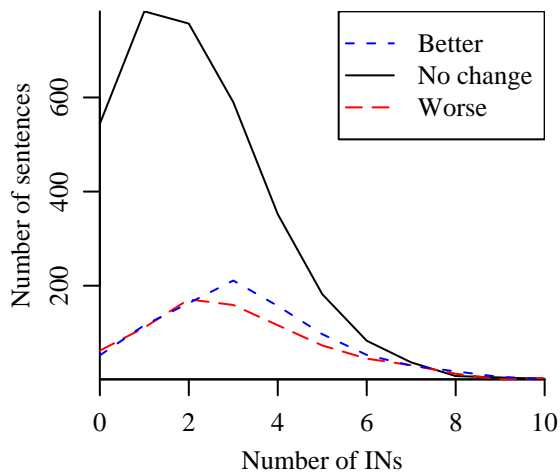In Table 5 the first column gives the name of the fac-

Figure 4: How self-training improves performance as a function of number of prepositions

tor. The second the change in the log-odds resulting from this factor being present (in the case of CCs and INs, multiplied by the number of them) and the last column is the probability that this factor is really non-zero.

Note that there is no row for either PPs or unknown words. This is because we also asked the program to do a model search using the Akaike Information Criterion (AIC) over all single and pairwise factors. The model it chooses predicts that the self-trained parser is likely produce a better parse than the baseline only for sentences of length 20–40 or sentences containing several CCs. It did not include the number of unknown words and the number of INs as factors because they did not receive a weight significantly different from zero, and the AIC model search dropped them as factors from the model.

In other words, the self-trained parser is more likely to be correct for sentences of length 20–40 and as the number of CCs in the sentence increases. The self-trained parser does *not* improve prepositional-phrase attachment or the handling of unknown words.

This result is mildly perplexing. It is fair to say that neither we, nor anyone we talked to, thought conjunction handling would be improved. Conjunctions are about the hardest things in parsing, and we have no grip on exactly what it takes to help parse them. Conversely, everyone expected improvements on unknown words, as the self-training should dras-
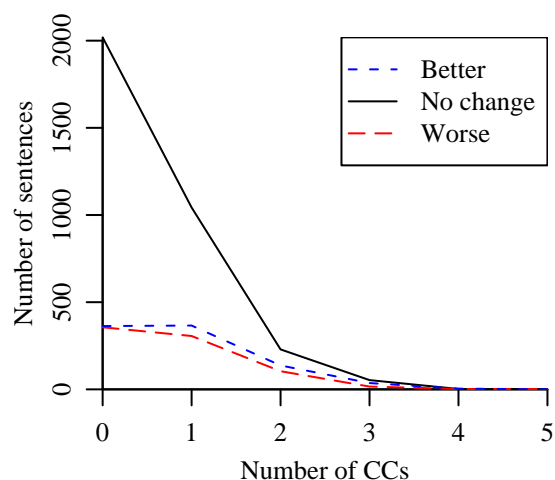
Figure 5: How self-training improves performance as a function of number of conjunctions

tically reduce the number of them. It is also the case that we thought PP attachment might be improved because of the increased coverage of preposition-noun and preposition-verb combinations that work such as (Hindle and Rooth, 1993) show to be so important.

Currently, our best conjecture is that unknowns are not improved because the words that are unknown in the WSJ are not significantly represented in the LA Times we used for self-training. CCs are difficult for parsers because each conjunct has only one secure boundary. This is particularly the case with longer conjunctions, those of VPs and Ss. One thing we know is that self-training always improves performance of the parsing model when used as a language model. We think CC improvement is connected with this fact and our earlier point that the probabilities of the 50-best parses are becoming more skewed. In essence the model is learning, in general, what VPs and Ss look like so it is becoming easier to pull them out of the stew surrounding the conjunct. Conversely, language modeling has comparatively less reason to help PP attachment. As long as the parser is doing it consistently, attaching the PP either way will work almost as well.

6 Conclusion

Contrary to received wisdom, self-training can improve parsing. In particular we have achieved an absolute improvement of 0.8% over the baseline performance. Together with a 0.3% improvement due to superior reranking features, this is a 1.1% improvement over the previous best parser results for section 23 of the Penn Treebank (from 91.0% to 92.1%). This corresponds to a 12% error reduction assuming that a 100% performance is possible, which it is not. The preponderance of evidence suggests that it is somehow the reranking aspect of the parser that makes this possible, but given no idea of why this should be, so we reserve final judgement on this matter.

Also contrary to expectations, the error analysis suggests that there has been no improvement in either the handing of unknown words, nor prepositional phrases. Rather, there is a general improvement in intermediate-length sentences (20-50 words), but no improvement at the extremes: a phenomenon we call the Goldilocks effect. The only specific syntactic phenomenon that seems to be affected is conjunctions. However, this is good news since conjunctions have long been considered the hardest of parsing problems.

There are many ways in which this research should be continued. First, the error analysis needs to be improved. Our tentative guess for why sentences with unknown words failed to improve should be verified or disproven. Second, there are many other ways to use self-trained information in parsing. Indeed, the current research was undertaken as the control experiment in a program to try much more complicated methods. We still have them to try: restricting consideration to more accurately parsed sentences as training data (sentence selection), trying to learn grammatical generalizations directly rather than simply including the data for training, etc.

Next there is the question of practicality. In terms of speed, once the data is loaded, the new parser is pretty much the same speed as the old — just under a second a sentence on average for treebank sentences. However, the memory requirements are largish, about half a gigabyte just to store the data. We are making our current best self-trained parser available[3] as machines with a gigabyte or more of RAM are becoming commonplace. Nevertheless, it would be interesting to know if the data can be pruned to

---

[3]`ftp://ftp.cs.brown.edu/pub/nlparser`

make the entire system less bulky.

Finally, there is also the nature of the self-trained data themselves. The data we use are from the LA Times. Those of us in parsing have learned to expect significant decreases in parsing accuracy even when moving the short distance from LA Times to Wall Street Journal. This seemingly has not occurred. Does this mean that the reranking parser somehow overcomes at least small genre differences? On this point, we have some pilot experiments that show great promise.

Acknowledgments

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98)*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan, June. Association for 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*, Menlo Park. AAAI Press/MIT Press.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *1st Annual Meeting of the NAACL*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*.

Stephen Clark, James Curran, and Miles Osborne. 2003. Bootstrapping POS-taggers using unlabelled data. In *Proceedings of CoNLL-2003*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the 17th International Conference (ICML 2000)*, pages 175–182, Stanford, California.

Sanjoy Dasgupta, M.L. Littman, and D. McAllester. 2001. PAC generalization bounds for co-training. In *Advances in Neural Information Processing Systems (NIPS), 2001*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

David Graff. 1995. *North American News Text Corpus*. Linguistic Data Consortium. LDC95T21.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proc. 42nd Meeting of Association for Computational Linguistics (ACL 2004), Barcelona, Spain*.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

Liang Huang and David Chang. 2005. Better k-best parsing. Technical Report MS-CIS-05-08, Department of Computer Science, University of Pennsylvania.

Victor M. Jimenez and Andres Marzal. 2000. Computation of the n best parse trees for weighted and stochastic context-free grammars. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*. Springer LNCS 1876.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *The Proceedings of the 37th Annual Conference of the Association for Computational Linguistics*, pages 535–541, San Francisco. Morgan Kaufmann.

Dan Klein and Christopher Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the ACL*.

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.

Anoop Sarkar. 2001. Applying cotraining methods to statistical parsing. In *Proceedings of the 2001 NAACL Conference*.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of EACL 03*.