Bootstrapping Statistical Parsers from Small Datasets

Mark Steedman[*], Miles Osborne[*], Anoop Sarkar[¶], Stephen Clark[*], Rebecca Hwa[§]
Julia Hockenmaier[*], Paul Ruhlen[†], Steven Baker[‡] and Jeremiah Crim[†]

[*]Division of Informatics, University of Edinburgh
{steedman,stephenc,julia,osborne}@cogsci.ed.ac.uk

[¶]School of Computing Science, Simon Fraser University anoop@cs.sfu.ca

[§]Institute for Advanced Computer Studies, University of Maryland hwa@umiacs.umd.edu

[†]Center for Language and Speech Processing, Johns Hopkins University
jcrim@jhu.edu,ruhlen@cs.jhu.edu

[‡]Department of Computer Science, Cornell University sdb22@cornell.edu

Abstract

We present a practical co-training method for bootstrapping statistical parsers using a small amount of manually parsed training material and a much larger pool of raw sentences. Experimental results show that unlabelled sentences can be used to improve the performance of statistical parsers. In addition, we consider the problem of bootstrapping parsers when the manually parsed training material is in a different domain to either the raw sentences or the testing material. We show that bootstrapping continues to be useful, even though no manually produced parses from the target domain are used.

1 Introduction

In this paper we describe how co-training (Blum and Mitchell, 1998) can be used to bootstrap a pair of statistical parsers from a small amount of annotated training data. Co-training is a weakly supervised learning algorithm in which two (or more) learners are iteratively retrained on each other's output. It has been applied to problems such as word-sense disambiguation (Yarowsky, 1995), web-page classification (Blum and Mitchell, 1998) and named-entity recognition (Collins and Singer, 1999). However, these tasks typically involved a small set of labels (around 2–3) and a relatively small parameter space. It is therefore instructive to consider co-training for more complex models. Compared to these earlier models, a statistical parser has a larger parameter space, and instead of class labels, it produces recursively built parse trees as output.

Previous work in co-training statistical parsers (Sarkar, 2001) used two components of a single parsing framework (that is, a parser and a supertagger for that parser). In contrast, this paper considers co-training two diverse statistical parsers: the Collins lexicalized PCFG parser and a Lexicalized Tree Adjoining Grammar (LTAG) parser.

Section 2 reviews co-training theory. Section 3 considers how co-training applied to training statistical parsers can be made computationally viable. In Section 4 we show that co-training outperforms self-training, and that co-training is most beneficial when the seed set of manually created parses is small. Section 4.4 shows that co-training is possible even when the set of initially labelled data is drawn from a different distribution to either the unlabelled training material or the test set; that is, we show that co-training can help in porting a parser from one genre to another. Finally, section 5 reports summary results of our experiments.

2 Co-training: theory

Co-training can be informally described in the following manner (Blum and Mitchell, 1998):

- Pick two (or more) "views" of a classification problem.
- Build separate models for each of these "views" and train each model on a small set of labelled data.
- Sample from an unlabelled data set to find examples for each model to label independently (Nigam and Ghani, 2000).
- Those examples labelled with high confidence are selected to be new training examples (Collins and Singer, 1999; Goldman and Zhou, 2000).
- The models are re-trained on the updated training examples, and the procedure is iterated until the unlabelled data is exhausted.

Effectively, by picking confidently labelled data from each model to add to the training data, one model is labelling data for the other. This is in contrast to self-training, in which a model is retrained only on the labelled examples that it produces (Nigam and Ghani, 2000).

Blum and Mitchell prove that, when the two views are *conditionally independent* given the label, and each view is sufficient for learning the task, co-training can improve an initial weak learner using unlabelled data. Dasgupta et al. (2002) extend the theory of co-training by showing that, by maximising their agreement over the unlabelled data, the two learners make few generalisation errors (under the same independence assumption adopted by Blum and Mitchell). Abney (2002) argues that this assumption is extremely restrictive and typically violated in the data, and he proposes a weaker independence assumption. Abney also presents a greedy algorithm that maximises agreement on unlabelled data.

Goldman and Zhou (2000) show that, through careful selection of newly labelled examples, co-training can work even when the classifiers' views do not fully satisfy the independence assumption.

3 Co-training: practice

To apply the theory of co-training to parsing, we need to ensure that each parser is capable of learning the parsing task alone and that the two parsers have different views. We could also attempt to maximise the agreement of the two parsers over unlabelled data, using a similar approach to that given by Abney. This would be computationally very expensive for parsers, however, and we therefore propose some practical heuristics for determining which labelled examples to add to the training set for each parser.

Our approach is to decompose the problem into two steps. First, each parser assigns a score for every unlabelled sentence it parsed according to some *scoring function*, f, estimating the reliability of the label it assigned to the sentence (e.g. the probability of the parse). Note that the scoring functions used by the two parsers do not necessarily have to be the same. Next, a *selection method* decides which parser is retrained upon which newly parsed sentences. Both scoring and selection phases are controlled by a simple incremental algorithm, which is detailed in section 3.2.

3.1 Scoring functions and selection methods

An ideal scoring function would tell us the true accuracy rates (e.g., combined labelled precision and recall scores) of the trees that the parser produced. In practice, we rely on computable scoring functions that approximate the true accuracy scores, such as measures of uncertainty. In this paper we use the probability of the most likely parse as the scoring function:

$$f_{prob}(\mathbf{w}) = \max_{v \in \mathcal{V}} Pr(v, \mathbf{w}) \qquad (1)$$

where $\mathbf{w}$ is the sentence and $\mathcal{V}$ is the set of parses produced by the parser for the sentence. Scoring parses using parse probability is motivated by the idea that parse probability should increase with parse correctness.

During the selection phase, we pick a subset of the newly labelled sentences to add to the training sets of both parsers. That is, a subset of those sentences labelled by the LTAG parser is added to the training set of the Collins PCFG parser, and vice versa. It is important to find examples that are reliably labelled by the *teacher* as training data for the *student*. The term *teacher* refers to the parser providing data, and *student* to the parser receiving

A and B are two different parsers.
M_A^i and M_B^i are models of A and B at step i.
U is a large pool of unlabelled sentences.
U^i is a small cache holding subset of U at step i.
L is the manually labelled seed data.
L_A^i and L_B^i are the labelled training examples for A and B at step i.
Initialise:
$L_A^0 \leftarrow L_B^0 \leftarrow L.$
$M_A^0 \leftarrow \mathit{Train}(A, L_A^0)$
$M_B^0 \leftarrow \mathit{Train}(B, L_B^0)$
Loop:
$U^i \leftarrow$ Add unlabeled sentences from U.
M_A^i and M_B^i parse the sentences in U^i and assign scores to them according to their scoring functions f_A and f_B.
Select new parses $\{P_A\}$ and $\{P_B\}$ according to some selection method S, which uses the scores from f_A and f_B.
L_A^{i+1} is L_A^i augmented with $\{P_B\}$
L_B^{i+1} is L_B^i augmented with $\{P_A\}$
$M_A^{i+1} \leftarrow \mathit{Train}(A, L_A^{i+1})$
$M_B^{i+1} \leftarrow \mathit{Train}(B, L_B^{i+1})$

Figure 1: The pseudo-code for the co-training algorithm

data. In the co-training process the two parsers alternate between teacher and student. We use a method which builds on this idea, $S_{\mathit{top}\text{-}n}$, which chooses those sentences (using the teacher's labels) that belong to the teacher's n-highest scored sentences.

For this paper we have used a simple scoring function and selection method, but there are alternatives. Other possible scoring functions include a normalized version of f_{prob} which does not penalize longer sentences, and a scoring function based on the entropy of the probability distribution over all parses returned by the parser. Other possible selection methods include selecting examples that one parser scored highly and another parser scored lowly, and methods based on disagreements on the label between the two parsers. These methods build on the idea that the newly labelled data should not only be reliably labelled by the teacher, but also be as useful as possible for the student.

3.2 Co-training algorithm

The pseudo-code for the co-training process is given in Figure 1, and consists of two different parsers and a central control that interfaces between the two parsers and the data. At each co-training iteration, a small set of sentences is drawn from a large pool of unlabelled sentences and stored in a *cache*. Both parsers then attempt to parse every sentence in the cache. Next, a subset of the sentences newly labelled by one parser is added to the training data of the other parser, and vice versa.

The general control flow of our system is similar to the algorithm described by Blum and Mitchell; however, there are some differences in our treatment of the training data. First, the cache is flushed at each iteration: instead of only replacing just those sentences moved from the cache, the entire cache is refilled with new sentences. This aims to ensure that the distribution of sentences in the cache is representative of the entire pool and also reduces the possibility of forcing the central control to select training examples from an entire set of unreliably labelled sentences. Second, we do not require the two parsers to have the same training sets. This allows us to explore several selection schemes in addition to the one proposed by Blum and Mitchell.

4 Experiments

In order to conduct co-training experiments between statistical parsers, it was necessary to choose two parsers that generate comparable output but use different statistical models. We therefore chose the following parsers:

1. The Collins lexicalized PCFG parser (Collins, 1999), model 2. Some code for (re)training this parser was added to make the co-training experiments possible. We refer to this parser as **Collins-CFG**.

2. The Lexicalized Tree Adjoining Grammar (LTAG) parser of Sarkar (2001), which we refer to as the **LTAG** parser.

In order to perform the co-training experiments reported in this paper, LTAG derivation events

Collins-CFG	LTAG
Bi-lexical dependencies are between lexicalized nonterminals	Bi-lexical dependencies are between elementary trees
Can produce novel elementary trees for the LTAG parser	Can produce novel bi-lexical dependencies for Collins-CFG
When using small amounts of seed data, abstains less often than LTAG	When using small amounts of seed data, abstains more often than Collins-CFG

Figure 2: Summary of the different views given by the Collins-CFG parser and the LTAG parser

were extracted from the head-lexicalized parse tree output produced by the Collins-CFG parser. These events were used to retrain the statistical model used in the LTAG parser. The output of the LTAG parser was also modified in order to provide input for the re-training phase in the Collins-CFG parser. These steps ensured that the output of the Collins-CFG parser could be used as new labelled data to re-train the LTAG parser and vice versa.

The domains over which the two models operate are quite distinct. The LTAG model uses tree fragments of the final parse tree and combines them together, while the Collins-CFG model operates on a much smaller domain of individual lexicalized non-terminals. This provides a mechanism to bootstrap information between these two models when they are applied to unlabelled data. LTAG can provide a larger domain over which bi-lexical information is defined due to the arbitrary depth of the elementary trees it uses, and hence can provide novel lexical relationships for the Collins-CFG model, while the Collins-CFG model can paste together novel elementary trees for the LTAG model.

A summary of the differences between the two models is given in Figure 2, which provides an informal argument for why the two parsers provide contrastive views for the co-training experiments. Of course there is still the question of whether the two parsers really are independent enough for effective co-training to be possible; in the results section we show that the Collins-CFG parser is able to learn useful information from the output of the LTAG parser.

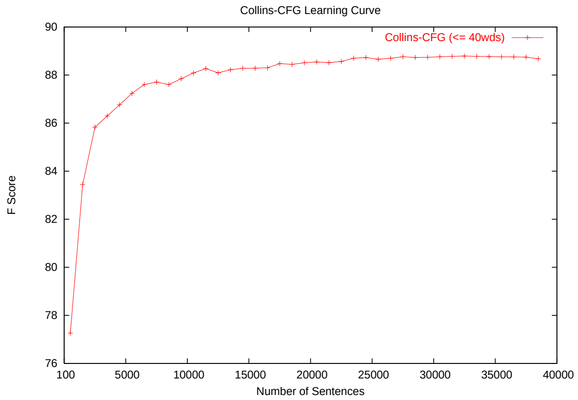

Figure 3: The learning curve for the Collins-CFG parser in terms of F-scores for increasing amounts of manually annotated training data. Performance for sentences ≤ 40 words is plotted.

4.1 Experimental setup

Figure 3 shows how the performance of the Collins-CFG parser varies as the amount of manually annotated training data (from the Wall Street Journal (WSJ) Penn Treebank (Marcus et al., 1993)) is increased. The graph shows a rapid growth in accuracy which tails off as increasing amounts of training data are added. The learning curve shows that the maximum payoff from co-training is likely to occur between 500 and 1,000 sentences. Therefore we used two sizes of seed data: 500 and 1,000 sentences, to see if co-training could improve parser performance using these small amounts of labelled seed data. For reference, Figure 4 shows a similar curve for the LTAG parser.

Each parser was first initialized with some labelled seed data from the standard training split (sections 2 to 21) of the WSJ Penn Treebank.

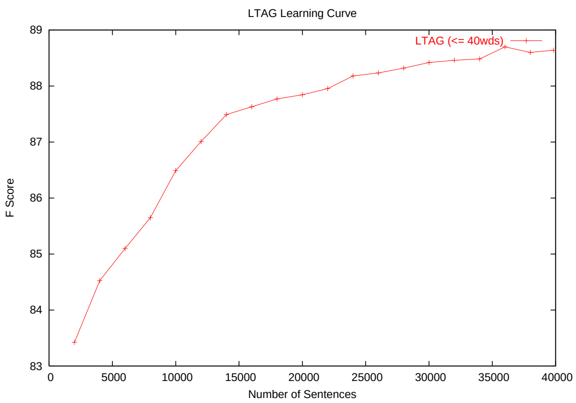

Figure 4: The learning curve for the LTAG parser in terms of F-scores for increasing amounts of training data. Performance when evaluated on sentences of length ≤ 40 words is plotted.

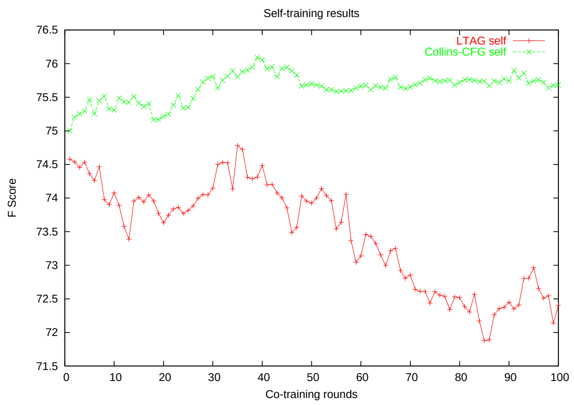

Figure 5: Self-training results for LTAG and Collins-CFG. The upper curve is for Collins-CFG; the lower curve is for LTAG.

Evaluation was in terms of Parseval (Black et al., 1991), using a balanced F-score over labelled constituents from section 0 of the Treebank.[1] The F-score values are reported for each iteration of co-training on the development set (section 0 of the Treebank). Since we need to parse all sentences in section 0 at each iteration, in the experiments reported in this paper we only evaluated one of the parsers, the Collins-CFG parser, at each iteration. All results we mention (unless stated otherwise) are F-scores for the Collins-CFG parser.

4.2 Self-training experiments

Self-training experiments were conducted in which each parser was retrained on its own output. Self-training provides a useful comparison with co-training because any difference in the results indicates how much the parsers are benefiting from being trained on the output of *another* parser. This experiment also gives us some insight into the differences between the two parsing models. Self-training was used by Charniak (1997), where a modest gain was reported after re-training his parser on 30 million words.

The results are shown in Figure 5. Here, both parsers were initialised with the first 500 sentences from the standard training split (sections 2 to 21) of the WSJ Penn Treebank. Subsequent unlabelled sentences were also drawn from this split. During each round of self-training, 30 sentences were parsed by each parser, and each parser was retrained upon the 20 self-labelled sentences which it scored most highly (each parser using its own joint probability (equation 1) as the score).

The results vary significantly between the Collins-CFG and the LTAG parser, which lends weight to the argument that the two parsers are largely independent of each other. It also shows that, at least for the Collins-CFG model, a minor improvement in performance can be had from self-training. The LTAG parser, by contrast, is hurt by self-training.

4.3 Co-training experiments

The first co-training experiment used the first 500 sentences from sections 2-21 of the Treebank as seed data, and subsequent unlabelled sentences were drawn from the remainder of these sections. During each co-training round, the LTAG parser parsed 30 sentences, and the 20 labelled sentences with the highest scores (according to the LTAG joint probability) were added to the training data of the Collins-CFG parser. The training data of the LTAG parser was augmented in the same way, using the 20 highest scoring parses from the set of 30, but using the Collins-CFG parser to label the sentences and provide the joint probability for scoring.

Figure 6 gives the results for the Collins-CFG parser, and also shows the self-training curve for

[1] F-score = $\frac{2 \times LR \times LP}{LR + LP}$ where LP is labelled precision and LR is labelled recall.

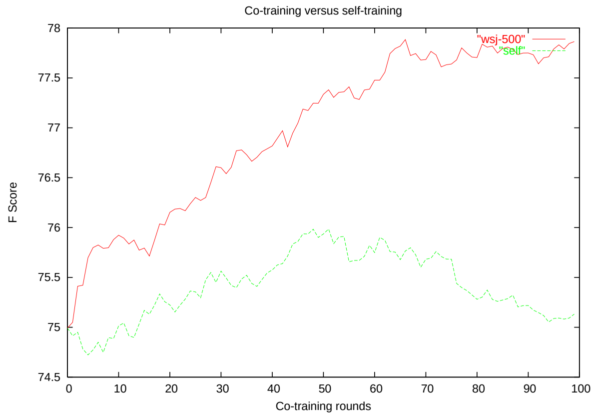

Figure 6: Co-training compared with self-training. The upper curve is for co-training between Collins-CFG and LTAG; the lower curve is self-training for Collins-CFG.

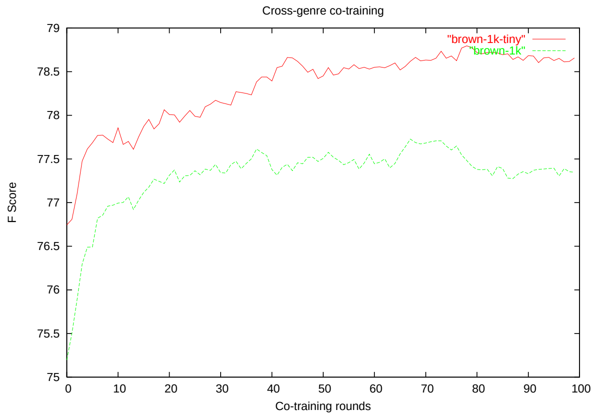

Figure 8: Cross-genre bootstrapping results. The upper curve is for 1,000 sentences labelled data from Brown plus 100 WSJ sentences; the lower curve only uses 1,000 sentences from Brown.

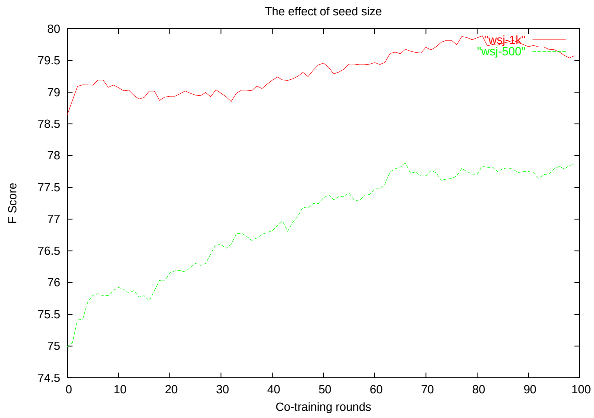

Figure 7: The effect of varying seed size on co-training. The upper curve is for 1,000 sentences labelled seed data; the lower curve is for 500 sentences.

comparison.[2] The graph shows that co-training results in higher performance than self-training. The graph also shows that co-training performance levels out after around 80 rounds, and then starts to degrade. The likely reason for this dip is noise in the parse trees added by co-training. Pierce and Cardie (2001) noted a similar behaviour when they co-trained shallow parsers.

[2]Figures 6, 7 and 8 report the performance of the Collins-CFG parser. We do not report the LTAG parser performance in this paper as evaluating it at the end of each co-training round was too time consuming. We did track LTAG performance on a subset of the WSJ Section 0 and can confirm that LTAG performance also improves as a result of co-training.

The second co-training experiment was the same as the first, except that more seed data was used: the first 1,000 sentences from sections 2-21 of the Treebank. Figure 7 gives the results, and, for comparison, also shows the previous performance curve for the 500 seed set experiment. The key observation is that the benefit of co-training is greater when the amount of seed material is small. Our hypothesis is that, when there is a paucity of initial seed data, coverage is a major obstacle that co-training can address. As the amount of seed data increases, coverage becomes less of a problem, and the co-training advantage is diminished. This means that, when most sentences in the testing set can be parsed, subsequent changes in performance come from better parameter estimates.

Although co-training boosts the performance of the parser using the 500 seed sentences from 75% to 77.8% (the performance level after 100 rounds of co-training), it does not achieve the level of performance of a parser trained on 1,000 seed sentences. Some possible explanations are: that the newly labelled sentences are not reliable (i.e., they contain too many errors); that the sentences deemed reliable are not informative training examples; or a combination of both factors.

4.4 Cross genre experiments

This experiment examines whether co-training can be used to boost performance when the unlabelled data are taken from a different source than the initial seed data. Previous experiments in Gildea (2001) have shown that porting a statistical parser from a source genre to a target genre is a non-trivial task. Our two different sources were the parsed section of the Brown corpus and the Penn Treebank WSJ. Unlike the WSJ, the Brown corpus does not contain newswire material, and so the two sources differ from each other in terms of vocabulary and syntactic constructs.

1,000 annotated sentences from the Brown section of the Penn Treebank were used as the seed data. Co-training then proceeds using the WSJ.[3] Note that no manually created parses in the WSJ domain are used by the parser, even though it is evaluated using WSJ material. In Figure 8, the lower curve shows performance for the Collins-CFG parser (again evaluated on section 0). The difference in corpus domain does not hinder co-training. The parser performance is boosted from 75% to 77.3%. Note that most of the improvement is within the first 5 iterations. This suggests that the parsing model may be adapting to the vocabulary of the new domain.

We also conducted an experiment in which the initial seed data was supplemented with a tiny amount of annotated data (100 manually annotated WSJ sentences) from the domain of the unlabelled data. This experiment simulates the situation where there is only a very limited amount of labelled material in the novel domain. The upper curve in Figure 8 shows the outcome of this experiment. Not surprisingly, the 100 additional labelled WSJ sentences improved the initial performance of the parser (to 76.7%). While the amount of improvement in performance is less than the previous case, co-training provides an additional boost to the parsing performance, to 78.7%.

5 Experimental summary

The various experiments are summarised in Table 1. As is customary in the statistical parsing literature, we view all our previous experiments using section 0 of the Penn Treebank WSJ as contributing towards development. Here we report on system performance on unseen material (namely section 23 of the WSJ). We give F-score results for the Collins-CFG parser before and after co-training for section 23.

[3]The Brown corpus was chosen as the seed data and the WSJ as the unlabelled data for convenience.

Experiment	Before	After
WSJ Self-training	74.4	74.3
WSJ (500) Co-training	74.4	76.9
WSJ (1k) Co-training	78.6	79.0
Brown co-training	73.6	76.8
Brown+ small WSJ co-training	75.4	78.2

Table 1: Results on section 23 for the Collins-CFG parser after co-training with the LTAG parser

The results show a modest improvement under each co-training scenario, indicating that, for the Collins-CFG parser, there is useful information to be had from the output of the LTAG parser. However, the results are not as dramatic as those reported in other co-training papers, such as Blum and Mitchell (1998) for web-page classification and Collins and Singer (1999) for named-entity recognition. A possible reason is that parsing is a much harder task than these problems.

An open question is whether co-training can produce results that improve upon the state-of-the-art in statistical parsing. Investigation of the convergence curves (Figures 3 and 4) as the parsers are trained upon more and more *manually-created* treebank material suggests that, with the Penn Treebank, the Collins-CFG parser has nearly converged already. Given 40,000 sentences of labelled data, we can obtain a projected value of how much performance can be improved with additional reliably labelled data. This projected value was obtained by fitting a curve to the observed convergence results using a least-squares method from `MATLAB`.

When training data is projected to a size of 400K manually created Treebank sentences, the performance of the Collins-CFG parser is projected to be 89.2% with an absolute upper bound of 89.3%. This suggests that there is very little room for performance improvement for the Collins-CFG parser by simply adding more labelled data (using co-training or other bootstrapping methods or even manually). However, models whose parameters have not already converged might benefit from co-training. For instance, when training data is projected to a size of 400K manually created Treebank sentences, the performance of the LTAG statistical parser would be 90.4% with an absolute upper bound of 91.6%. Thus, a bootstrapping method might improve performance of the LTAG statistical parser beyond the current state-of-the-art performance on the Treebank.

6 Conclusion

In this paper, we presented an experimental study in which a pair of statistical parsers were trained on labelled and unlabelled data using co-training. Our results showed that simple heuristic methods for choosing which newly parsed sentences to add to the training data can be beneficial. We saw that co-training outperformed self-training, that it was most beneficial when the seed set was small, and that co-training was possible even when the seed material was from another distribution to both the unlabelled material or the testing set. This final result is significant as it bears upon the general problem of having to build models when little or no labelled training material is available for some new domain.

Co-training performance may improve if we consider co-training using *sub-parses*. This is because a parse tree is really a large collection of individual decisions, and retraining upon an entire tree means committing to all such decisions. Our ongoing work is addressing this point, largely in terms of re-ranked parsers. Finally, future work will also track comparative performance between the LTAG and Collins-CFG models.

Acknowledgements

This work has been supported, in part, by the NSF/DARPA funded 2002 Language Engineering Workshop at Johns Hopkins University. We would like to thank Michael Collins, Andrew McCallum, and Fernando Pereira for helpful discussions, and the reviewers for their comments on this paper.

References

Steven Abney. 2002. Bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 360–367, Philadelphia, PA.

E. Black, S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 306–311.

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*, pages 92–100, Madisson, WI.

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

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Empirical Methods in NLP Conference*, pages 100–110, University of Maryland, MD.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Sanjoy Dasgupta, Michael Littman, and David McAllester. 2002. PAC generalization bounds for co-training. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the Empirical Methods in NLP Conference*, Pittsburgh, PA.

Sally Goldman and Yan Zhou. 2000. Enhancing supervised learning with unlabeled data. In *Proceedings of the 17th International Conference on Machine Learning*, Stanford, CA.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 9th International Conference on Information and Knowledge Management*, pages 86–93.

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *Proceedings of the Empirical Methods in NLP Conference*, Pittsburgh, PA.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the 2nd Annual Meeting of the NAACL*, pages 95–102, Pittsburgh, PA.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA.