CLSP WS-02 Final Report: Semi-Supervised Training for Statistical Parsing

Mark Steedman,[*] Steven Baker,[†] Jeremiah Crim,[+]
Stephen Clark,[*] Julia Hockenmaier,[*] Rebecca Hwa,[#]
Miles Osborne,[*] Paul Ruhlen,[+] Anoop Sarkar[♭]
([*]Edinburgh, [♭]Penn, [#]UMD, [+]JHU, [†]Cornell)

April 7, 2003

CLSP WS-02 Final Report: Semi-Supervised Training for Statistical Parsing

Mark Steedman,[*] Steven Baker,[†] Jeremiah Crim,[+]
Stephen Clark,[*] Julia Hockenmaier,[*] Rebecca Hwa,[#]
Miles Osborne,[*] Paul Ruhlen,[+] Anoop Sarkar[♭]
([*]Edinburgh, [♭]Penn, [#]UMD, [+]JHU, [†]Cornell)

April 7, 2003

1 Introduction

This project investigated co-training (Blum and Mitchell 1998) as a method for bootstrapping wide-coverage parsers initially trained on hand-labeled data. Hand-labeled data-sets—even the 1M-word Penn Wall Street Journal Treebank—are never large enough to train parsers effectively, and are expensive to produce. Automated or semi-automated methods for cheaply generating more labeled training data are an interesting alternative.

Co-training is a technique originally developed for classifiers, and uses the output of two or more classifiers, initially trained on hand-labeled data, then run on (much more plentiful) unlabeled data to provide informative additional training data for each other. Crucially, the classifiers must have different models or "views" of the data to provide additional labeled data for training each other.

Applying co-training to parsers raises a number of questions, since parsing is more than simple classification using a small finite set of labels. However, Sarkar (2001) showed that co-training could be used to improve performance of the LTAG parser trained on a subset of the Penn Wall Street Journal Treebank, with the TAG equivalent of a POS tagger representing the alternate view. The present project was proposed in order to investigate the conditions under which such co-training effects for parsers can be reliably obtained, and how they can be maximized.

Such techniques for semi-automatically increasing the amount of training data available for training parsers are of considerable interest. Even the 1M words of the Penn WSJ corpus is arguably not enough to train adequate models, and for other languages and even other genres of English text, one is going to have to manage with much less than that.

Among the possible payoffs of reliable methods for co-training parsers are therefore the following:

- Improved performance of existing wide-coverage parsers
- Methods for building very large treebanks larger and less noisy than the 30M word BLLIP corpus of output from Charniak's 1997 parser, and useful for numerous speech/language applications
- Methods for bootstrapping parsers for novel genres and new languages from small labeled datasets

1.1 Statistical parsing: the State of the Art

The project was motivated by the following observations. Parsers trained on the Penn Wall Street Journal Treebank have improved rapidly, and have been widely and usefully applied. These applications would be even more numerous and effective if those parsers could be be trained on or ported to other genres of text in English and other languages. However, annotated corpora of a size comparable to the Penn Wall Street Journal's 1M words in any *other* genre or language will always be the exception.

Nor does "self-training"—training the parser on its own output—offer a way out. Charniak (1997) showed a small improvement from retraining his parser on 30M words of its own output, probably because the increased size of the training set gave somewhat better counts for the head dependency model. However, training on data that merely confirms a parser's existing view cannot help in any other way, and one might expect such improvements to be transient and eventually to deteriorate in the face of otherwise noisy data.

However, it is known that ensemble techniques using information from different models via techniques like voting can improve results over any one view alone (Henderson and Brill 1999), and the result already mentioned from Sarkar (2001) suggests that co-training across the different views might be a useful alternative.

1.2 What is Co-training?

Co-Training (Blum and Mitchell 1998) is a weakly supervised method for bootstrapping a model from a relatively small seed set of labeled examples, augmented by a much larger set of unlabeled examples by exploiting redundancy among multiple statistical models. Crucially, in contrast to Self-training, it involves training on another model's output. The basic procedure is as follows:

- A set of models is trained on labeled seed material.

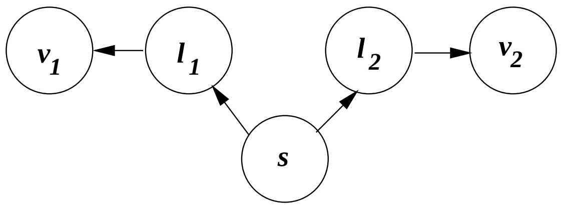


Figure 1: Dasgupta et al. 2002

- The whole set of models is then run on a larger amount of unlabeled data. Novel labeled examples from any model that are deemed reliable under some measure are used as additional labeled data for retraining the other models.

- The previous step is iterated with the retrained models run on more unlabeled data.

Co-training can be thought of theoretically as seeking to optimize an objective function that measures the degree of agreement between the predictions for the unlabeled data based on the two views (Blum and Mitchell 1998; Collins and Singer 1999). Dasgupta et al. 2002 define the conditional probability relationships in figure 1.2, where v_i is a view, l_i is a label, and s is a sentence. By the Data Processing Inequality, $I(v_1;s) \geq I(v_1;v_2)$, if views v_1 and v_2 agree a lot, they tell us a lot about the observation s. Nigam and Ghani (2000) analyze the independence assumption. Dasgupta et al. provide a PAC-style justification.

We noted above that the theory of co-training has been developed in application to classifiers, and that its application to parsers raises a number of questions to be discussed as the paper proceeds. In particular, optimizing the objective function can only be approximated using heuristics to guess which novel analyses are reliable for training other parsers, using measures like *Intersection* (choose those of the top n other's output that are in your bottom n) and *Difference* (choose those of the other's outputs for which their score is higher than yours by some margin). Factors such as the two parsers' confidence in their respective analyses suggest a possible basis for these heuristics, among other possibilities that are investigated below.

1.3 Project Details

The project looked at various co-training pairs across three distinct groups of parsers, namely:

1. Treebank CFG parsers (Collins 1999, 2001);

2. The Lexicalized Tree-Adjoining Grammar (LTAG) parser (Sarkar (2001));

3. Combinatory Categorial Grammar (CCG) parsers and "supertaggers" (Clark 2002; Hockenmaier and Steedman 2002b:

(A supertagger is the equivalent of a part of speech (POS) tagger for CCG or TAG grammars, but models a set of lexical types (larger by an order of magnitude) that the more expressive grammars assume.)

One of the problems in co-training existing parsers is that their output formats are different and not necessarily intertranslatable: for example, CCG has a much larger vocabulary of nonterminal symbols than TAG, and both CCG and TAG include information equivalent to traces, while Collins and Charniak do not. We therefore only investigated certain combinations of parsers, rather than all those theoretically possible. In particular the project investigated the following pairs:

1. Co-training with supertaggers and parsers for the same grammar type (CCG)

2. Co-training with different parsers (CFG and LTAG)

3. Co-training Re-Rankers for a single parser (Collins 2000)

We will report these three groups of experiments separately, drawing some common conclusions later. All of the experiments use some or all of the hand-annotated data of the Penn Treebank as the initial labeled set. In most cases this is the 1M-word Wall Street Journal section. The exception is the experiments on "porting" a trained parser to a new genre using unlabeled data in the new genre: in this case we trained on (some or all) of the 440K-word Brown Corpus section of the treebank and ported to the WSJ material, rather than vice versa. This was in order to be able to compare the effectiveness of porting co-training against the effects of same-genre co-training experiments using the (larger and more reliable) WSJ sections. In the case of parsers trained on the entire WSJ section, we used unlabeled data from the 500M-word North American News Text corpus. However, for smaller experiments using only a subset of the WSJ corpus as labeled data, we used the sentences (but not the trees) of the remainder of sections 2-22 as our unlabeled data: this was again for reasons of calibration, and in order to compare the effects of co-training with those of training on human-labeled data. All testing was done on the held-out section 0 of the WSJ treebank.

1.4 Progress Before the Workshop

In preparation for the workshop, a considerable amount of prior work was necessary. Once the project was accepted and the personnel agreed (we asked to split the standard stipends and expenses over five postdoctoral level researchers rather than three in order to encompass a sufficient diversity of parsing expertise), and having established a mailing list and a collaborative web-page, we began to work as a team by email and via a series of four meetings interactive in the period May-July 2002. Because of the transatlantic nature of the group, only the first of these (in Baltimore) and the last (at the ACL conference in Philadelphia just before the workshop) were physical. The intervening two meetings (at which most of the work was done) were via videoconferencing. These meetings were extremely successful, and we recommend the technique to similarly geographically extended groups in future workshops. We found that the slight reduction in communicative immediacy was more than compensated for by the reduction in jetlag and general wear and tear on the participants.

These meetings were the driving force behind the following preparatory exercises:

1. Designing and implementing a common framework and basic architecture for all co-training experiments, including unlabeled data cache management, and accepting parser output from all frameworks.

2. Cleaning, tokenizing, and tagging the 500M words of News corpus unlabeled data, removing large quantities of SGML cruft and transduction artefacts.

3. Matching all parsers including imported ones to the datasets, including News and Brown corpora.

4. Variously tweaking and rewriting all parsers for retraining, re-ranking, recognizing each others outputs etc.

5. Parsing substantial amounts of unlabeled data in preparation for large cache self-training and co-training benchmarks for all parsers.

6. One iteration of parsing of Brown corpus in preparation for porting experiment for most parsers.

7. Pilot experiments showing co-training effects for two versions of the Collins 1999 parser trained on artificially constructed subsets of WSJ data, one excluding all sentences with prepositional phrases and one excluding those with coordination, to induce different views.

1.5 Progress During the Workshop

Our goals for the workshop itself were the following:

1. To identify criteria for parser output selection that exclude noise and maximize co-training effects;

2. To explore the way co-training effects depend on the size of labeled seed set;

3. To explore effectiveness of co-training for porting parsers to new genres by training on Brown Corpus, co-training on unlabeled WSJ and using held-out PT-WSJ labeled secn. 00 for testing.

4. Explore effectiveness of co-training on parsers trained on all of PT-WSJ 2-21, co-trained on unlabeled WSJ and tested on labeled secn. 00

The experiments we describe in the sections that follow will demonstrate the following results.

1. The experiments show that co-training enhances performance for parsers and taggers trained on small (500-10,000 sentences) amounts of labeled data—that is, for labeled datasets of the kind of size that can realistically be expected to be obtainable at short notice for novel languages and novel genres of text.

2. The experiments also show that co-training can be used for porting parsers trained on one genre to parse on another without any new human-labeled data at all.

3. The experiments also show that even tiny amounts of human-labeled data for the target genre enhance porting via co-training.

4. Distinguishing reliable and informative newly labeled data from less reliable and informative output is crucial. The research developed a number of novel methods for parse selection.

5. The experiments also yield some preliminary results on ways to deliver similar improvements for parsers trained on large (Penn WSJ Treebank) labeled datasets and expressive grammars such as TAG and CCG.

1.6 Progress Since the Workshop

Since the end of the workshop, the group has continued to collaborate, with the following outcomes to date:

1. The results have been written up in a number of conference submissions, of which five have already been accepted, including Steedman et al. 2003a,b, which report the core results of the workshop itself (See Appendix B, Publications Arising from the Workshop Project).

2. In subsequent work, Clark et al. (2003) looked at directly maximising agreement between two POS taggers. This could not be done for our parsers, as noted above, but was investigated using parser output reranking (section 5 below). Baldridge and Osborne (2003) show that similar techniques provide a basis for an active learning system for HPSG parse selection. Callison-Burch and Osborne (2003) apply the techniques for co-training developed in the workshop to the task of statistical machine translation. At the time of writing, two further papers are in preparation: the first (Sarkar et al. 2003) investigates corrected co-training for parsers. The second (Osborne et al. 2003) considers the relationship between EM and co-training of statistical parsers.

3. As a result of preliminary results using a perceptron to directly implement optimization of the objective function for parse-re-ranking under item 4 above, a successful proposal for support from the workshop for a continuation project was made by Miles Osborne and Jay Crim. The latter will be visiting Edinburgh this summer to continue work on parse re-ranking.

4. A successful proposal for a CLSP Summer Worshop project for 2003 has been made by Anoop Sarkar as part of a group including Dan Gildea.

1.7 Structure of the Report

The next section discusses the basic co-training architecture and data management issues, and the important issue of parse selection. We then present three series of experiments. In section 3, the first set of experiments examines the effect of using output from the LTAG parser to co-train Collins' treebank CFG, and the effect of using Collins' output to co-train the LTAG parser. In section 4, a second set of experiments looks for co-training effects across the CCG parser and the CCG-Supertagger, and includes an investigation of parser output selection using the the sentences of the treebank as unlabeled material and using the parseval measure as an "Oracle" to select parses for co-training under various criteria. In section 5, a third set of experiments examines the use of co-training to train various re-rankers for a re-ranking parser based on Collins 2001. A concluding section brings these results together and summarizes.

2 Co-training for statistical parsing

Co-training and their close variants have traditionally been applied to classification tasks such as word sense disambiguation (Yarowsky (1995)), named entity identification (Collins (1999)), and web page classification (Blum and Mitchell (1998)). In this work, we propose a framework for co-training between multiple parsers, in which the labeled output produced by each parser is the result of combining many local classification decisions. Because parsing is a complex and computationally expensive process in and of itself, a major challenge is to adapt co-training for parsing such that the co-training process remains computationally tractable. In this section, we first describe the architecture of our framework; then we discuss the theoretical and practical considerations in adapting co-training for statistical parsing; to meet the challenges of the parser training task, we propose novel methods for selecting newly labeled examples (i.e., parse trees produced by the learner parsers); finally, we present two oracle experiments to test the feasibility of our ideas. The results suggest that finding a good selection method may be important for parsers to co-train effectively.

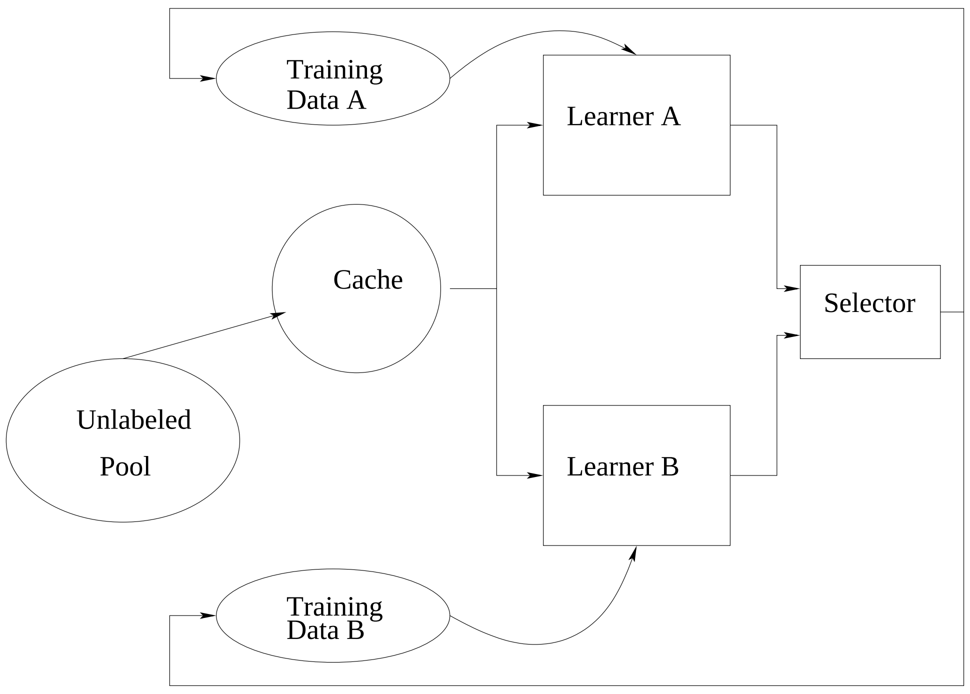


Figure 2: Diagram of our co-training framework.

2.1 Architecture

The architecture of our system is illustrated in Figure 2. It consists of two *different*[1] parsers and a central control that interfaces between the two parsers and the data, called the *Cache Manager*. At each co-training iteration, the Cache Manager randomly draws a small set of sentences from a large pool of unlabeled sentences and stores them in the *cache*. Both parsers then attempt to parse every sentence in the cache[2]. Next, the Cache Manager selects a subset of the newly labeled sentences[3] according to some objective function and adds them to

[1]We shall discuss the notion of difference further in section 2.2.

[2]In our system, the parsers are n-best statistical parsers that output multiple parse trees (and assign a probability value to each tree) for each sentence.

[3]The *label* that a parser assigns to a sentence is the parse tree with the highest probability. This notion of label is somewhat different from other classification tasks because a parse tree is made up of multiple sub-components. It would be interesting to consider a finer-grained notion of examples rather than entire sentences, though this is outside the scope of the work reported

```
$A$ and $B$ are two different parsers.
The Cache Manager ($CM$) is the central control that
  interfaces with both parsers.
$U$ is a large pool of unlabeled sentences.
$U'$ is a small cache holding a subset of $U$.
$L_i$ is the set of labeled training examples for parser $i$.
$H_i$ is the current hypothesis of parser $i$.
**Initialize:**
    $L_A = L_B \leftarrow L$.
    $H_A \leftarrow Train(L_A)$
    $H_B \leftarrow Train(L_B)$
**Loop**
    $CM$ moves unlabeled sentences from $U$ to $U'$.
    $A$ and $B$ parses the sentences in $U'$ and assigns reliability scores
        to them according to some scoring function $f$.
    $CM$ selects data to add to $L_A$ and $L_B$ according to
        some selection method $S$.
    $H_A \leftarrow Train(L_A)$
    $H_B \leftarrow Train(L_B)$
```

Figure 3: The pseudo-code for the co-training learning algorithm

the training sets of the parsers. The pseudocode for the co-training process is sketched out in Figure 3.

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 ensures that the distribution of sentences in the cache is representative of the entire pool. Also, this reduces the possibility of forcing the Cache Manager to choose training examples from an entire set of unreliably labeled 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

here.

the one proposed by Blum and Mitchell. (We shall discuss selection methods in more detail in section 2.3).

2.2 Theoretical and Practical Considerations

Central to the co-training algorithm is the idea that the two learners must be sufficiently different from each other such that even though each is capable of learning the task by itself, they can help each other to learn faster in tandem. Each learner is said to have a different *view* of the learning task, and co-training is the process in which the learners try to optimize for agreements on labeling the unlabeled data.

How different must the views be to satisfy the "sufficiently different" criterion? Theoretical work by Blum and Mitchell (1998) and Dasgupta et al. (2002) proves that co-training works when the views are *conditionally independent* given the label. For example, suppose we have two learners that try to differentiate basketballs from baseballs; Learner A does so by the color of the object, and Learner B by size. The two views of the learners are conditionally independent because if Learner A knows the classification of the object, information about the size of the object is irrelevant (and similarly, information about the color of the object is irrelevant for Learner B once the classification is known). Recent work by Abney (2002) suggests that view independence may be too strong and proposes a more relaxed assumption of rule independence. Abney has also sketched an algorithm for finding classifiers that agree on unlabeled data. In essense, the algorithm performs a greedy search: for every potential update to the learners, the algorithm evaluates how well the two learners would agree on some set of unlabeled data after the update; then the update with the best score would be actually carried out.

To apply the theory of co-training to parsing, we need to ensure that the two parsers are sufficiently different, that each parser is capable of learning the parsing task alone, and that we can explicitly optimize an *objective function* that measures the degree of agreement between the two parsers' predictions for the unlabeled sentences. We will discuss the first two criteria in more details when we present our main experiments using different the parser pairs in later sections. The last criterion poses a practical difficulty for the parsing task. To explicitly maximize agreement in the manner proposed by Abney, we would need to evaluate how well the two parsers would agree (on a set of N unlabeled heldout sentences, say) after being retrained on each newly labeled sentence in the cache (of size C). In order to pick just one sentence to add to the parsers' training set, the two parsers would have to be run $C \times N$ times. Because this is computationally expensive, in the next section, we propose some practical heuristics for determining which labeled examples to add to the training set for each parser.

2.3 Selection of labeled examples

Our approach is to decompose the problem into two steps. First, each parser assigns a score for every unlabeled sentence it parsed according to some *scoring function*, f, estimating the reliability of the label it assigned to the sentence (i.e., the most likely parse). Note that the scoring functions used by the two parsers do not necessarily need to be the same. In section 2.3.1 we propose some possible scoring functions. Next, the Cache Manager uses the scores for the sentences from both parsers to select the newly labeled sentences to add to each parser's training set according to some *selection method*, S. We propose some possible selection methods in section 2.3.2.

2.3.1 Scoring functions: An ideal scoring function would tell us the true accuracy rates (e.g., combined labeled precision and recall scores[4]) of the trees that the parser produced. In practice, we rely on computable scoring functions that approximates the true accuracy scores, such as measures of uncertainty[5]. One possible scoring function is to use the probability of the most likely parse. The intuition is to trust the parser's judgment: if it assigned a higher probability value to the label of one sentence than that of another, then the label for this sentence is deemed more reliable. The function is denoted as:

$$f_{prob}(\mathbf{w}) = Pr(v_{max}, \mathbf{w}) = Max_{v \in V} Pr(v, \mathbf{w}).$$

where $\mathbf{w}$ is the sentence, v_{max} is the parse tree with the highest probability out of the set of all the parses produced by the parser for the sentence, V.

Because f_{prob} uses the joint probability of the parse tree and the sentence, it favors shorter sentences[6]. To directly compare the probabilities between sentences of different lengths, we define $f_{norm\text{-}prob}$, a scoring function based on

[4]A commonly used metric is the *F-score*, which is defined as $\frac{1}{\frac{\alpha}{LP} + \frac{(1-\alpha)}{LR}}$, where LR is the labeled recall score and LP is the labeled precision score, and α determines the weighting of precision and recall. Typically, $\alpha = 0.5$, such that $\frac{2 \times LR \times LP}{LR + LP}$.

[5]The role of the scoring functions is similar to those used by Hwa (2000) for sample selection.

[6]This may not be problematic since the parses for longer sentences do tend to be less accurate.

conditional probabilities:

$$f_{norm\text{-}prob}(\mathbf{w}) = Pr(v_{max} \mid \mathbf{w}) = \frac{Pr(v_{max}, \mathbf{w})}{\sum_{v \in V} Pr(v, \mathbf{w})}$$

Focusing solely on the most likely parse may not be sufficient to characterize the parser's uncertainty. For example, suppose we wish to compare one sentence for which the parser generated four equally likely parses with another sentence for which the parser generated one parse with probability of 0.2 and ninety-nine other parses with probabilities in range of 0.01. Even though the parser's output for the second sentence is more certain, f_{prob} would assign a higher score to the first sentence. We consider another scoring function, $f_{entropy}$, which takes into account the probability distribution of all parses by comparing the entropy of the distribution to that of a uniform distribution.

$$f_{entropy}(\mathbf{w}) \quad = \quad \frac{-\sum\limits_{v \in V} Pr(v \mid \mathbf{w}) \lg(Pr(v \mid \mathbf{w}))}{\lg(\|V\|)}.$$

Spikier distributions have lower entropy, suggesting that the parser is more certain about its label for that sentence.

One might also consider some coarse-grained metrics such as the number of parses produced (denoted as f_{nparse}) and the length of the sentence (denoted as f_{len}) because they are easy to compute.

2.3.2 Selection Methods: During the selection phase, the Cache Manager picks a subset of the newly labeled sentences from the cache to add to the training set of both parsers. In this section, we describe several selection methods that try to find examples with *minimal noise* and *maximal training utility*. First, we consider a baseline selection method (denoted as S_{base}) that picks labeled examples with the n-highest scores. This method is similar to the selection process in Blum and Mitchell (1998) in that the training sets for the two parsers are identical; that is, they contain examples labeled by both parsers.

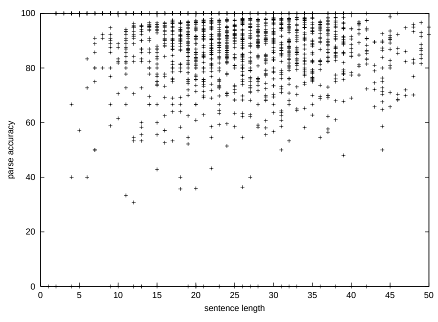
For the rest of the selection methods that we propose, the examples added to the training set of one parser (referred to as the *student*) are only those labeled by the other (referred to as the *teacher*). For these selection methods, the Cache Manager first treats one parser as the student and the other the teacher, then reverses their roles. Analogous to the baseline, $S_{top\text{-}n}$ chooses the examples for which the teacher assigned the n-highest scores. Both S_{base} and $S_{top\text{-}n}$ focus on minimizing noise. In order to also maximize training utility, it is important to find examples that are reliably labeled by the teacher but unreliably labeled by the student. One such selection method is $S_{intersect}$, which chooses those sentences (using the teacher's labels) that belong to the intersection of the teacher's *n*-highest scored sentences and the student's *n*-lowest scored sentences. Note that $S_{intersect}$ does not make use of the absolute scores the parsers assigned to their outputs; the selection is determined by the relative rankings alone. Therefore, this selection method would not be suitable for situations in which the student's label is less reliable than the teacher (i.e., the score is lower) but the relative ranking is similar to that of the teacher. Moreover, $S_{intersect}$ focuses on the extreme cases in which the teacher's *most* reliably labeled sentences are also the student's *least* reliably labeled sentences. It might be sufficient to only require the teacher's label to be more reliable than the students. Following the less stringent criterion, we propose a method that selects examples based on score differences, called S_{diff}. A sentence would be chosen if the teacher assigned a higher score to its label than the student by some threshold. Finally, we consider a method, $S_{disagree}$, that, in addition to score differences, also selects examples based on the degree of disagreement between the teacher's label and the student's label.

2.4 Oracle experiments

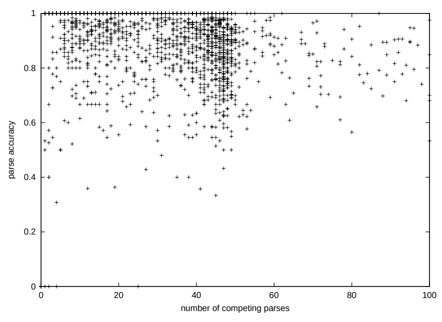
We have conducted two oracle experiments to investigate the feasibility of our approach. The first experiment studies the effect of different scoring functions. The second experiment studies the effect of different selection methods.

2.4.1 Scoring functions: In this study, we compare different scoring functions on their ability to predict the reliability of the labels (i.e., most likely parse trees) produced by the parser. In other words, we wish to see whether there exists some correlation between the true accuracy rates and the scores calculated by the proposed functions. To do so, we have used a trained CFG parser to parse a set of about 2,000 unseen test sentences. Reliability scores for the parser's output are computed according to each scoring functions. We then plot them against the true parse accuracy of these sentences. Ideally, we would like the scatter plot to show something that resembles a linear relationship between the values produced by the scoring functions (x-axis) and the parsing accuracies (y-axis).

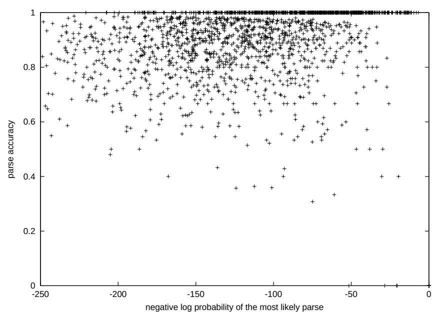
Figure 4 presents a qualitative comparison between the different scoring functions. None of the proposed scoring function exhibits strong linear rela-



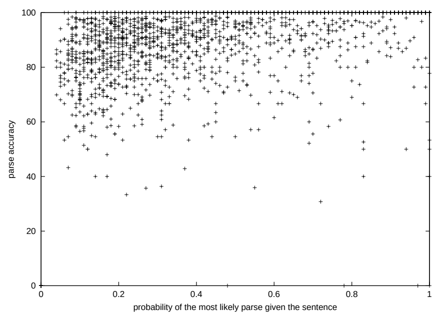
(a)



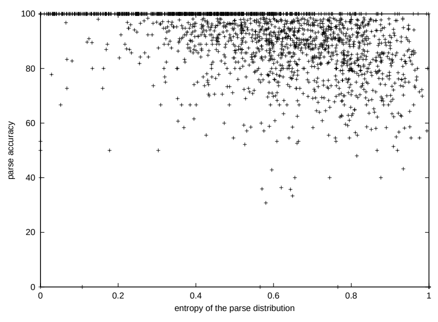
(b)



(c)



(d)



(e)

Figure 4: (a)Correlation of the sentence length and the parse accuracy of a sentence($fscore(\mathbf{w})$). (b)Correlation of the number of parses generated for a sentence and its accuracy rate. (c) Correlation of the joint probability of the best parse and the sentence and its parsing accuracy. (d) Correlation of the conditional probability of the best parse given the sentence and the parsing accuracy. (e) Correlation plot between $f_{entropy}(\mathbf{w})$ and $fscore(\mathbf{w})$

tionship to the accuracy rate, but some plots suggest some mild correlations. Of the five, f_{prob} (Figure 4(c)) and $f_{entropy}$ (Figure 4(e)) seem the most promising. Sentences with high f_{prob} scores or low $f_{entropy}$ scores were typically parsed correctly. Although the converse does not hold (i.e., low f_{prob} scores and high $f_{entropy}$ scores do not tell us much about the parse accuracy of the sentences), for the purpose of co-training, it is more important to be able to confidently identify the set of sentences whose parses are mostly accurate (e.g., a parsing F-score of 90% or higher) rather than differentiating between the highly inaccurately parsed sentences and the somewhat inaccurately parsed sentences. Moreover, assuming that we have no lack of unlabeled sentences, it may not be critical for the function to identify *all* accurately parsed sentences. That is, in the precision-recall trade-off for identifying reliably labeled sentences, we would want to favor precision over recall.

We have also conducted an experiment comparing the scoring functions as a part of the co-training framework. The set-up for this co-training experiment is somewhat non-traditional: we use two versions of the same parser that are trained on different initial seed data (one contains no prepositional phrase constructs, the other contains no conjunction constructs). In Figure 5, we plot the performances for one parser under different scoring functions. The selection method used for all cases is $S_{top\text{-}n}$.

2.4.2 Selection method variation: In this study, we investigate the effects of different selection methods on the co-training process. To remove the complication of noisy reliability scores affecting the selection methods, we have assumed that the parsers have access to an oracle that returns the true accuracy rate for their outputs. Note that the oracle scoring function does not change the parsers' output; that is, the labeled examples added to the parsers' training sets still contain errors.

We have conducted the study co-training a CFG parser (Collins (1999)) and an LTAG parser (Sarkar (2002)). Both parsers are initially trained on 1000 seed labeled sentences from the WSJ Treebank (Marcus et al. (1993)). At each co-training iteration, the Cache Manager fills the cache with 500 sentences from a pool of about 37,000 unlabeled sentences (Section 02-21 of the WSJ Treebank with the annotation stripped). The four proposed selection methods are compared to the baseline selection method[7] and an upper bound of training from

[7]For both S_{base} and $S_{top\text{-}n}$, the threshold for inclusion is set at an accuracy rate of 90%. For S_{diff}, the difference threshold is set to be 10%. For $S_{disagree}$, the disagreement threshold is

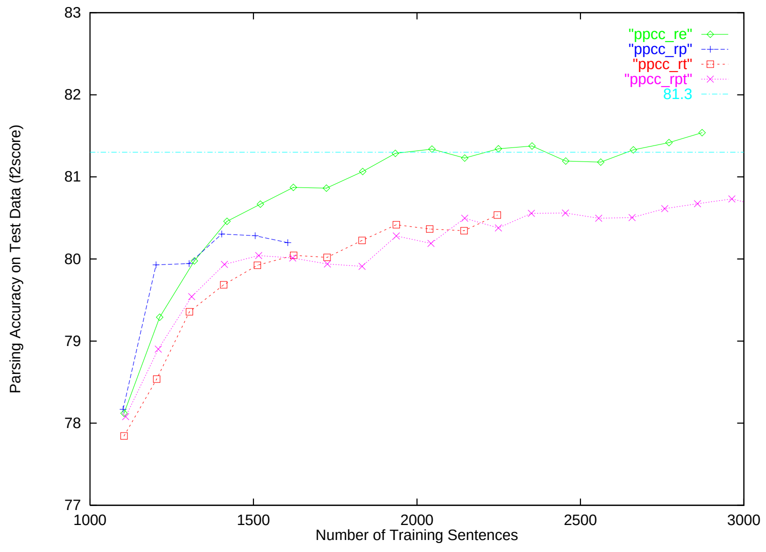


Figure 5: Scoring variation study

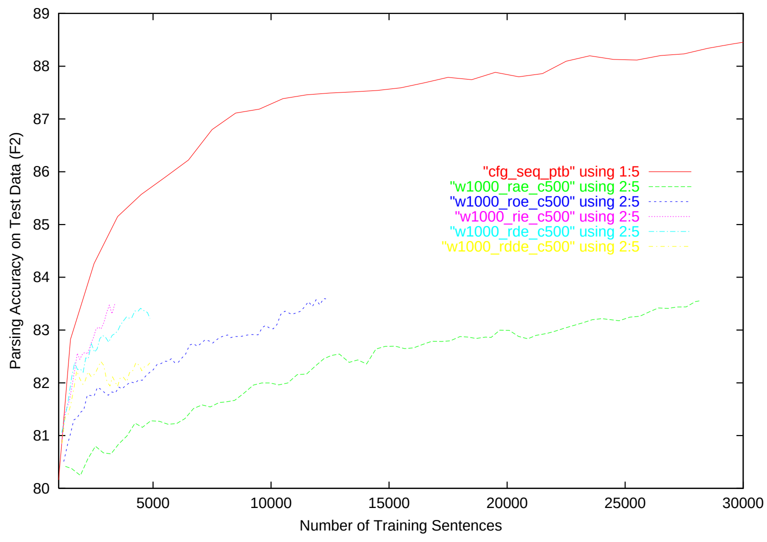


Figure 6: Selection variation oracle study.

set to be 30%. Finally, $S_{intersect}$ compares the top 40% of the sentences parsed by the teacher with the bottom 40% of those by the student.

hand-annotated data.

Both parsers have been evaluated; here, we report the results for the CFG parser, which received higher Parseval scores. Figure 6 plots the parser's progress in learning under different selection methods. The x-axis of the graph shows the number of sentences used in the training set and the y-axis of the graph shows the trained CFG parser's accuracy rate on unseen test sentences (Section 00 of the WSJ Treebank). The curves in the graphs show the rate of improvement for parsers trained using different selection methods. The curves have different ending points because some selection methods reject more sentences so that they exhaust the pool of unlabeled data sooner.

Three out of the four proposed selection methods help the CFG parser to learn faster than the baseline; however, none rivals the hand-annotated upper bound. With the exception of $S_{disagree}$, the selection methods that try to maximize training utility have an immediate effect on improving the parser's performance. In terms of the final performance level of the trained parser, however, it is not clear from this experiment whether they have a clear advantage over the others. For instance, while a parser trained under $S_{intersect}$ can achieve a parsing performance score of 83.5% after adding fewer than 3500 sentences, it would eventually achieve the same performance level under S_{base} after adding 28000 sentences. It may be the case that the proposed selection methods may have more of an advantage if the unlabeled pool is larger, or if the cache is replenished with some of the previously rejected sentences. It may also be the case that 83.5% is the upper limit of the parser's performance under the co-training regime (i.e., trained with labeled examples that are not error-free).

2.5 Section summary

In this section, we have described our co-training framework for inducing statistical parsers. In particular, we have have proposed several scoring functions for predicting the parser's accuracy rate and selection methods for choosing reliably labeled training examples. We have conducted oracle experiments to test the feasibility of these ideas.

In the scoring function oracle experiment, we found that two of the proposed functions, f_{prob} and $f_{entropy}$, could reliably identify some sentences with accurate parses; however, the functions are not able to approximate parsing accuracy rates for arbitrary sentences. Developing better predictors to find reliable parser output is a part of our on-going research effort.

The result of our selection method variation experiment suggests that those methods that consider both parsers' accuracy rates (that is, to select examples for which the teacher-parser is believed to have labeled better than the student-parser) seem to help the parsers to co-train more effectively. In our oracle study, the best performing selection method is $S_{intersect}$; however, because the performance of the selection method depends on the reliability of the scores the parsers assigned to their outputs, this conclusion may not generalize to our approximation scoring functions. In the next section, we conduct large-scale studies that will explore this issue further.

3 Co-training between Collins-CFG and LTAG

In order to conduct co-training experiments between statistical parsers, it was necessary to choose two parsers that generate comparable outputs but are different from each other in their statistical modeling details. In this section of the report, we consider using the following parser pair:

1. The Collins parser Collins (1999): We used the parser described in Mike Collins' PhD thesis which is available for download from the author's web page. Some code for (re)training this parser was added prior to the workshop to make the co-training experiments possible. We refer to this parser in our discussions below as the **Collins-CFG** parser since it uses a formalism closely related to probabilistic context-free grammars (PCFGs).

2. The LTAG parser Sarkar (2001): The second parser used was a parser based on the lexicalized tree-adjoining grammar (LTAG) formalism. This statistical parser was written by Anoop Sarkar and was used previously in parsing the Treebank and in some earlier co-training experiments using LTAG. We refer to this parser as the **LTAG** parser.

We chose this pair of parsers because they have comparable performance but are based on different grammar formalisms. In Section 3.1 we describe the two parsers in some detail and give both theoretical arguments and empirical evidence showing that the two parsers are sufficiently different for co-training. As mentioned earlier, our co-training framework affords a number of parameters (e.g., the size of the seed data, selection methods). In Section 3.2, we discuss the parameter choices we have made and the experimental setup for co-training with the Collins-CFG and LTAG parser pair. A suite of experiments is conducted using different parameter settings. In Section 3.3, we report the results of these

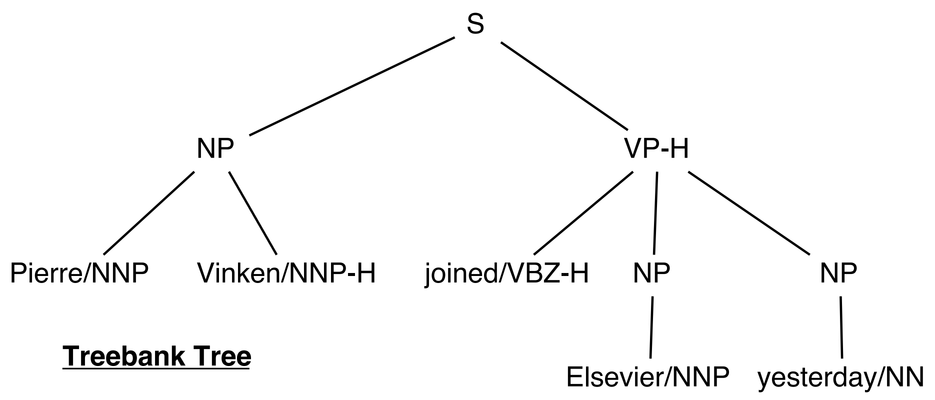


Figure 7: Treebank tree

experiments. Our findings suggest that co-training between the Collins-CFG and LTAG parsers is the most helpful when the initial seed data is small.

3.1 Description of the statistical parsers

While the two parsers are different statistical models, they share a similar input-output behavior. Both require training data that were annotated in the style of the Penn Treebank parse trees and augmented with head information. The head annotation was heuristically determined using standard head percolation rules (cf. Magerman (1994)). All trees used in our experiments (manually produced or automatically created) were annotated in this manner.

Figure 7 shows one such (manually produced) Treebank parse tree which is used for training the statistical parsers. The head information is marked with the -*H* marker on the non-terminal. This head information is used to lexicalize the parse tree, by passing the head word to the parent from a single child with the -*H* mark. In this way, each non-terminal is decorated with a single word from the input. Additional information about arguments vs. adjuncts is also gleaned from the Treebank tree and used in the parsing models. Lexical information plays a key part in the success of the probability models in each of the statistical parsers under consideration. As will become clear later, the methods in which the lexical information percolates through parse trees are different for the two parsers.

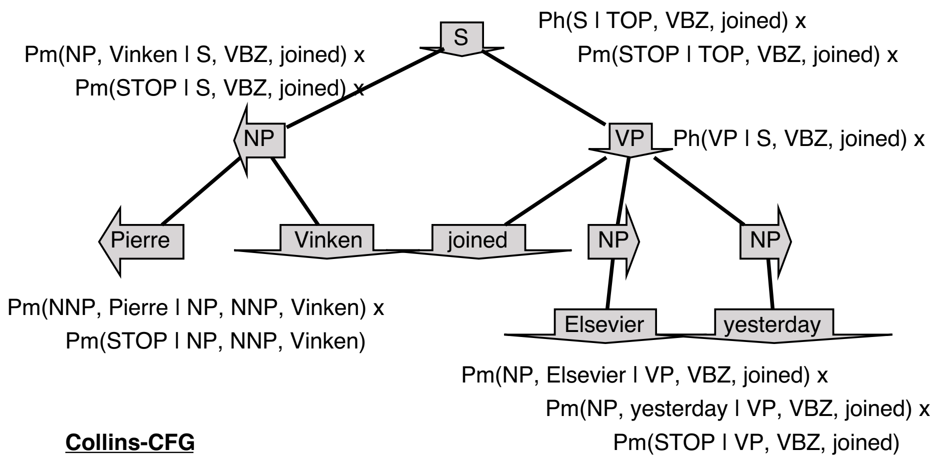

S
Ph(S | TOP, VBZ, joined) x
Pm(STOP | TOP, VBZ, joined) x
Pm(NP, Vinken | S, VBZ, joined) x
Pm(STOP | S, VBZ, joined) x
NP
VP
Ph(VP | S, VBZ, joined) x
Pierre
Vinken
joined
NP
NP
Pm(NNP, Pierre | NP, NNP, Vinken) x
Pm(STOP | NP, NNP, Vinken)
Elsevier
yesterday
Pm(NP, Elsevier | VP, VBZ, joined) x
Pm(NP, yesterday | VP, VBZ, joined) x
Pm(STOP | VP, VBZ, joined)
Collins-CFG

Figure 8: Collins-CFG derivation of a treebank tree.

3.1.1 Collins-CFG parser: The Collins-CFG parser splits up a parse tree into a series of lexicalized CFG rules, which are themselves split up into a sequence of decisions that make up each rule as pairs of lexicalized non-terminals. We illustrate the model by means of an example. From the Treebank tree shown in Figure 7, one particular lexicalized CFG rule that can be extracted is `VP`(*joined*) $\rightarrow$ `joined/VBZ-H NP`(*Elsevier*) `NP`(*yesterday*). Since this rule could have an arbitrary right-hand side it can have an arbitrary number of words leading to severe sparse data problems. The Collins-CFG model decomposes such a rule into a series of relations between lexicalized non-terminals as shown in Figure 8. The model is often referred to as *bi-lexical* since each pair of non-terminals leads to a bigram relationship between two (possible non-adjacent) words from the input. These bi-lexical parameters provide the statistical parser the means to choose the most plausible tree for the input sentence from the alternative parses. The probability models used are:

- P_h – the head percolation probability model, in which the head daughter of a parent node is selected. For example, in Figure 8, *VP* is selected as the head daughter of *S* and *S* also inherits the lexical information from *VP*.
- P_m – the modifiers on either side of the head daughter are generated using this probability model. This model relates pairs of words. For example in

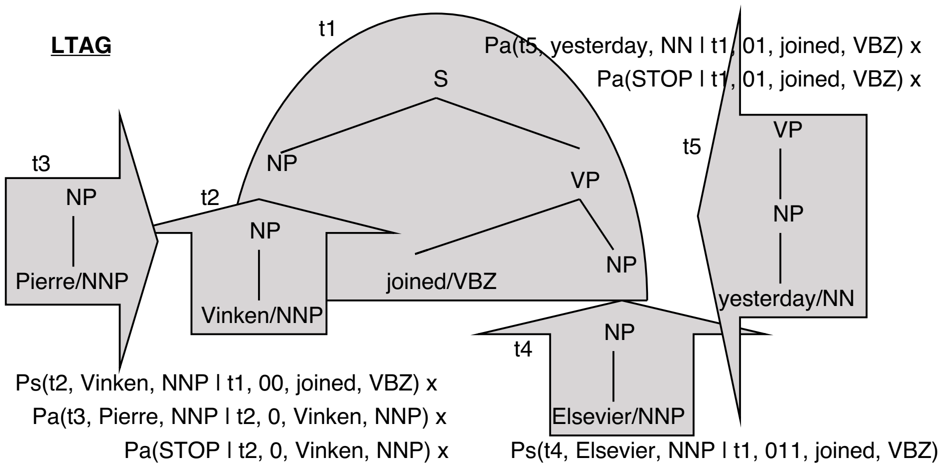

LTAG
t1
S
NP
VP
NP
joined/VBZ
t3
NP
Pierre/NNP
t2
NP
Vinken/NNP
t4
NP
Elsevier/NNP
t5
VP
NP
yesterday/NN
Pa(t5, yesterday, NN | t1, 01, joined, VBZ) x
Pa(STOP | t1, 01, joined, VBZ) x
Ps(t2, Vinken, NNP | t1, 00, joined, VBZ) x
Pa(t3, Pierre, NNP | t2, 0, Vinken, NNP) x
Pa(STOP | t2, 0, Vinken, NNP) x
Ps(t4, Elsevier, NNP | t1, 011, joined, VBZ)

Figure 9: LTAG derivation of a treebank tree

Figure 8, the first modifier to the right of the *VP(joined)* is *NP(Elsevier)*. After all the modifiers a special non-terminal called *STOP* is produced to keep the probabilities well-formed. In the Collins-CFG model, left and right modifiers are distinguished and have distinct probability models.

Additional information is also used in the Collins-CFG model such as adjacency and subcategorization information.

3.1.2 LTAG parser: The LTAG parser also relies on bigrams of lexical information in order to determine the most plausible parse tree just as in the Collins-CFG parser. However, the domain over which these bigrams are established are different from that of the Collins model. Instead of lexical relationships being mediated via non-terminals, these relations are established between trees, where each tree has a word from the input as a leaf. Such *lexicalized* trees are termed *elementary trees*. The final parse tree is created by rewriting non-terminals by these elementary trees. Non-terminals that are replaced with trees are both on the frontier of other trees or even non-leaf nodes like internal or root nodes. Figure 9 shows how the original Treebank tree is decomposed into elementary trees (trees t_1 through t_5) and indicates how the trees combine to form the original parse. The probability models used are:

- P_s – each non-terminal at the frontier is re-written by an elementary tree using the *substitution* probability: P_s. For example, in Figure 9 the tree

t_2 lexicalized by the word *Vinken* is substituted into the tree t_1 at the node labeled by *NP* specified by the unique specifier of a node in a tree (the Gorn address 00). The probability of this substitution is given by:

$$P_s(t_2, \text{Vinken}, \text{NNP} \mid t_1, 00, \text{joined}, \text{VBZ})$$

- P_a – Non-leaf nodes like internal or root nodes can be re-written by an elementary tree using the *adjunction* probability: P_a. This operation has a separate model since adjunction in an elementary tree is optional unlike substitution. For example in Figure 9, the modifier tree t_5 lexicalized by *yesterday* adjoins into the *VP* node (Gorn address 01) of the tree t_1 lexicalized by *joined*. The probability of this adjunction is given by:

$$P_a(t_5, \text{yesterday}, \text{NN} \mid t_1, 01, \text{joined}, \text{VBZ})$$

  As in the Collins-CFG model, left and right modifiers are distinguished and have distinct probability models.

3.1.3 Differences between Collins-CFG and LTAG parsing models: As can be seen by the descriptions of the two models, the domain 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 non-terminals. This provides a mechanism to bootstrap information between these two models when they are applied to unlabeled 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 Collins-CFG can paste together novel elementary trees for the LTAG model.

A summary of all the distinctions that the two models can use in order to bootstrap information for each other is provided in Figure 10. This table provides an argument that we can use the Collins-CFG and the LTAG models as contrastive views in a co-training experiment for statistical parsing. Note that this argument is informal, in that we have not proved that the Collins and LTAG models are truly statistically independent from each other. In fact, it is probably true that the two models are only partially independent of each other. For non-trivial situations (where models are developed with knowledge of each other), this partial dependence will almost certainly always be true.

Collins-CFG	LTAG
Bi-lexical dependencies are between nonterminals	Bi-lexical dependencies are between elementary trees
Can produce novel elementary trees for the LTAG	Can produce novel bi-lexical dependencies for Collins-CFG
Learning curves show convergence on 1M words labeled data	Learning curves show LTAG needs relatively more labeled data
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 10: Summary of the different views between the Collins-CFG parser and the LTAG parser

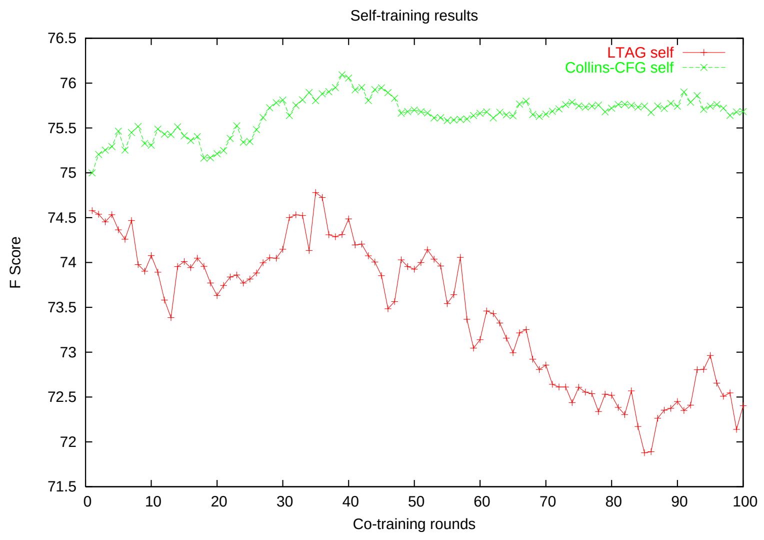


Figure 11: Comparison of self-training between Collins-CFG and LTAG

3.1.4 Experimental comparison of the difference between Collins-CFG and LTAG: To gain some experimental insight into the differences between the two statistical parsing models, we have conducted a self-training experiment with each parser. Self-training is a limited case of co-training, in that a model is only trained on the labeled examples that it produces (Nigam and Ghani 2000). In the context of our parsing experiments, this meant that the LTAG parser was re-trained upon sentences that were parsed by the LTAG parser. The Collins parser was retrained in a similar manner. Self-training was used in Charniak 1997, where a modest gain was reported after re-training his parser on 30 million un-parsed words.

The results are shown in Figure 11. Here, both parsers were initialized with the first 500 sentences from the standard training split (sections 2 to 21) of the parsed Wall Street Journal corpus. Subsequent disjoint unlabeled sentences were also drawn from this split. Evaluation was in terms of an F-score over labeled constituents. At each self-training round, 30 sentences were processed. As one would expect from parsing models that are distinct in their behavior with respect to the same input, the results of self-training vary significantly between the Collins-CFG and the LTAG parser. This empirical behavior lends weight to the argument that our two parsers are largely independent of each other. It also shows that at least for the Collins model, some benefit can be had from self-training. The LTAG parser by contrast, is hurt by self-training.

3.2 Experimental Setup

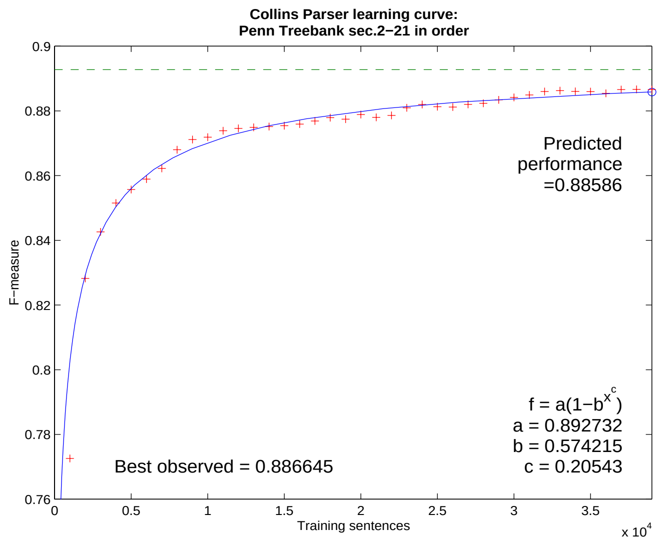
To determine how well suited the Collins-CFG and LTAG parser pair is for co-training, we have designed a suite of experiments aimed at achieving the stated workshop goals. We have varied three parameters. First, we have investigated the effect of varying the size of the initial seed data. In particular, we have considered using 500 and 1000 annotated sentences; Section 3.2.1 provides an experimental justification for our choices. Second, we have compared different selection methods (using f_{prob} as the scoring function). Based on the results of the oracle study we focus on $S_{intersect}$ and $S_{use\text{-}all}$. We use a cache size of 30 sentences per iteration, and $S_{intersect}$ selects the top 67% of the sentences labeled by the teacher that are in the bottom 67% of the sentences labeled by the student. Finally, we have studied the effect that changing corpus domain might have on co-training performances. More specifically, rather than assuming both the initial seed data and the unlabeled data were from the same source, we have looked at a scenario in which the unlabeled data are sentences from the Wall Street Journal corpus, but the initial annotated data are taken from the Brown corpus. We have also considered the case in which a small amount of sentences from the domain of the unlabeled data (e.g. 100 sentences) were hand-annotated and added to the seed data.

3.2.1 Size of the initial seed data: Statistical parsers are usually only evaluated after having been trained upon all available annotated material. This is only part of the story. What is also of crucial interest is the *convergence rate*. That is, how much training material is required to achieve a given level of performance. Knowing this behavior will tell us whether a given parser has already converged using standard training sets. So, we can tell whether a given parser would benefit from extra training material. It also tells us when we would likely see the best improvement (maximum rate of change) for co-training.

To this end, we plotted learning curves for both the Collins-CFG and the LTAG parser (See Figure 12). This was done to find the optimum amount of seed data for these kinds of parsing models, where co-training would provide the maximum payoff. The graphs show a rapid growth of accuracy which tails off as increasing amounts of training data is added. It should be noted that while the increase in accuracy between 30K and 40K sentences is minimal, to obtain the best accuracy the entire training data is required. As one can see from the learning curve, the maximum payoff occurs between 500 and 1000 sentences. Based on this experiment, we set our small seed data size to be 500 and 1000 sentences on which we start our experiments with the hypothesis that co-training will help improve accuracy over what is possible simply with this labeled data set size.

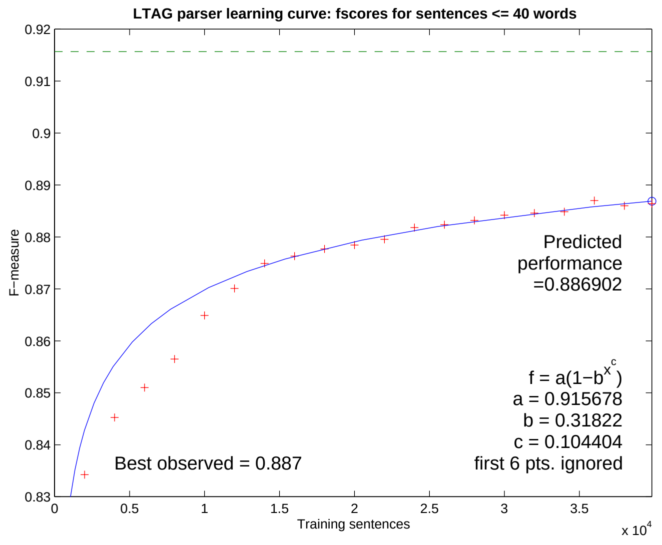
Extrapolating from the curve for the Collins-CFG parser, we see that the upper bound on performance is an f-score of 89.3 (on section 0). This performance would be obtained using about 40 thousand cleanly labeled sentences[8]. The addition of more labeled data even if it were labeled by humans would improve the accuracy of the Collins parser only by a negligible amount. This suggests that novel bi-lexical counts from additional data effect a corresponding increase in accuracy for the Collins parser. In contrast, the curve for the LTAG parser suggests that the addition of more labeled data (for instance, with the use of bootstrapping methods such as co-training) will improve LTAG parsing results.

[8]Note that a re-ranked version of this parser, without extra training material, does better than this already.



Upper bound = 89.3% Projected to Treebank of size 80K sentences = 88.9% 400K = 89.2%

(a)



Upper bound = 91.6% Projected to Treebank of size 80K sentences = 89.3% 400K = 90.4%

(b)

Figure 12: Learning curves. Accuracy plotted against the amount of training data used. (a) Collins-CFG parser. (b) LTAG parser.

3.3 Results and Discussions

We now present the experimental results of co-training using the Collins-CFG and LTAG parser pair. Before we examine how different parameters affect co-training, we present a baseline performance comparison between co-training and self-training. Figure 12 shows the performance curves for the Collins parser and the LTAG parser. Evaluation is in terms of an F-score with respect to section 0 of the treebank. In this baseline study, we used 500 sentences from the WSJ corpus as the initial seed data (same domain as the unlabeled data pool). The selection method used is $S_{use\text{-}all}$[9]. The graph shows that co-training results in higher performance than self-training. The graph also shows that co-training performance levels-out around 80 rounds, and then starts to degrade. The likely reason for this dip is noise in the co-training-supplied parse trees. Pierce and Cardie (2001) noted a similar behavior when they co-trained shallow parsers.

3.3.1 Varying initial seed data size: One of the workshop goals is to see whether co-training could boost performance when faced with small seed data sets. Figure 13 plots the performance curves of the Collins-CFG parser when co-training from initial seed data set sizes of 500 and 1000 sentences. Note that different selection methods were used for the two curves (we chose the best performing selection method for each seed data setting). It is interesting that the choice of selection method depends upon the seed size. We shall return to this point in Section 3.3.2 when we discuss the variation on selection method. The key observation from this experiment is that the benefit of co-training seems more dramatic 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's redundant views can address. As the amount of seed data increases, coverage becomes less of an issue, and the co-training advantage is diminished. Recall that another one of the workshop goals was to improve the state-of-art in statistical parsing using co-training. Based on our seed data variation experiment and the convergence curves presented earlier, we conjecture that the parsers have almost converged already. As such, extra training material are unlikely to yield a significant improvement for these parsers. The state-of-the-art in statistical parsing may improve if we use more sophisticated models that have not already converged (with respect to available treebank material). Within the co-training setting, one such approach is through re-ranking models. Another observation

[9]Note that graphs show performance after every round of co-training. This means that at the origin we have already carried out a single round.

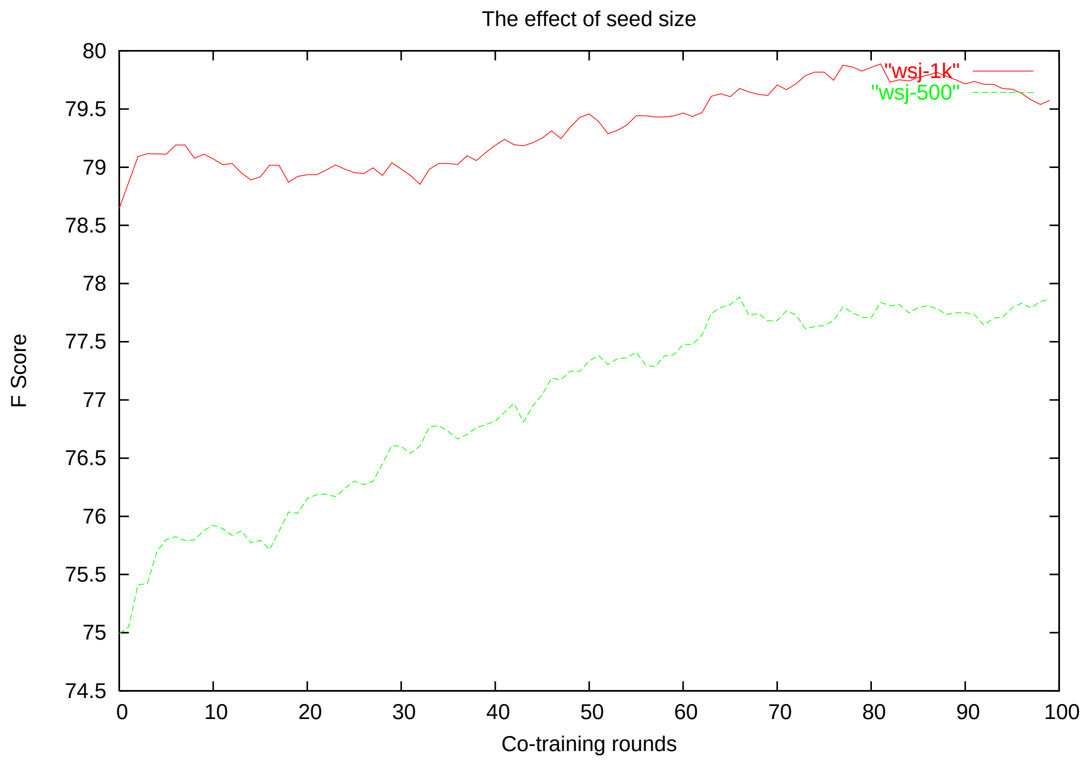


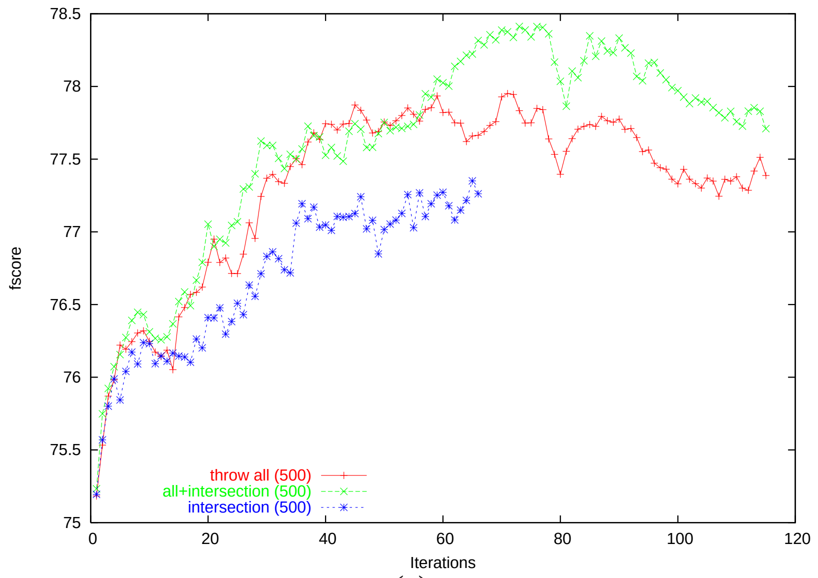
Figure 13: Performance comparison of different initial seed data sizes for the Collins-CFG parser.

is that although co-training boosts the performance of the parser using the 500 seed sentences from 75.3% to 78.4%, it does not achieve the level of performance of a parser trained on 1000 seed sentences. Some possible explanations include: the newly labeled sentences are not reliable (i.e., they contain too many errors); the sentences deemed reliable are not informative training examples; or a combination of both factors.

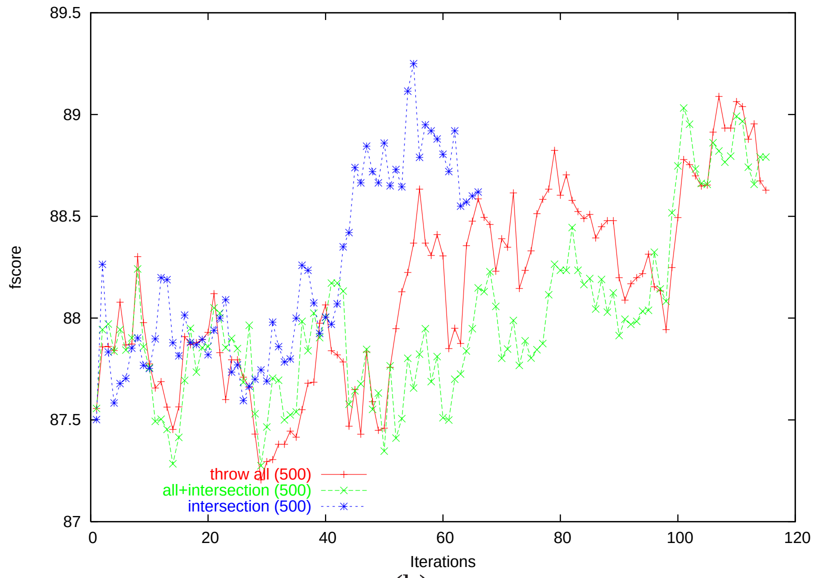
3.3.2 Comparison of parse selection methods: As we have argued earlier, the question of how one should select examples in practical co-training settings is important. That is, for computational reasons, we cannot afford to directly optimize agreement between views. Instead, we must approximate view agreement through example selection. Here, we present the *in vivo* counterpart to the oracle study. This study compares two selection methods: $S_{use\text{-}all}$, which indiscriminantly selects all newly labeled sentences, and $S_{intersect}$, which selects sentences that the teacher-parser labeled with high scores (probability of the parse in this experiment) and the student-parser labeled with low scores.

The curves in Figure 14 show co-training results for both the Collins-CFG parser and the LTAG parser under different selection methods. Both parsers are initially trained on 500 seed sentences. The key observation from the pair of graphs is that while $S_{use\text{-}all}$ results in better performance for the Collins-CFG parser, $S_{intersect}$ results in better performance for the LTAG parser. On the other hand, recall from the previous experiment that for the Collins-CFG parser, $S_{intersect}$ results in better performance when it is initially trained with 1000 seed data. One possible reason for these differences may be due to the trade-off between competing factors of increasing coverage and reducing noise. In the Collins-CFG case, increasing coverage is the more important factor when the seed data is very small. When starting from a larger seed data set that already has a good coverage, training on reliably labeled data is more important. Another possible reason is the inherent differences in the parsing models. Because the LTAG parser builds parse trees out of elementary trees, which provide larger structural context than CFG rules, the parse tree it generates may help the Collins-CFG parser to improve coverage even if it (the parse tree) contains many errors. On the other hand, reliability seems to be a more important factor for the LTAG parser. Noisy parse trees generated by the Collins parser may create too many poor elementary trees for the LTAG parser.

3.3.3 Cross corpus domain experiments: training on the Brown corpus and testing on WSJ: This experiment examines whether we can use co-training to boost performance when the unlabeled data are taken from a different source



(a)



(b)

Figure 14: Comparison between the *use-all* and the *intersection* selection methods. (a) The performance of the Collins-CFG parser. (b) The performance of the LTAG parser.

than the initial seed data. We have used 1000 annotated sentences from the Brown treebank (non-newswire material) as the seed data. The two parsers then co-train off of the Wall Street Journal (newswire material). In Figure 15, the lower curve shows performance for the Collins-CFG parser (evaluated, as ever, on section 0) using the intersection selection method. The difference in corpus domain does not hinder co-training. The parser performance is boosted from 76.6% to 78.3%. 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 have also conducted an experiment in which the initial seed data is supplemented with a tiny amount of annotated data (say 100 sentences) in the domain of the unlabeled data. The upper curve in Figure 15 shows the outcome of this experiment. Not surprisingly, the 100 additional labeled WSJ sentences improved the initial performance of the parser to 78.7%. While the amount of improvement in performance is less dramatic than the previous case, co-training provides an additional boost to the parsing performance, to 80%. Another observation of interest is that the parser improves gradually. This suggests that the parsers are learning novel constructs as well as vocabularies.

3.4 Summary for co-training between Collins-CFG and LTAG

In this section, we have explored co-training between two statistical parsers that are based on different grammar formalisms. We have conducted a suite of experiments varying the amount of initial seed data, the selection methods, and the corpus domain of the seed data. Our experimental results suggest that co-training between these two parsers can generally boost parsing performances. We find that the improvement is more dramatic under impoverished conditions such as when the amount of seed data is small or when the seed data is taken from a different corpus. We also find that different selection methods may work better under different conditions, depending on the parsing model and the amount of seed data as well as the scoring function.

4 Co-training a CCG Parser and Supertagger

This section describes co-training experiments using parsers from another grammar formalism, namely Combinatory Categorial Grammar (CCG, Steedman 2000). CCG is a lexicalized formalism, in which words and constituents are associated with **categories** which express their subcategorization behavior, and in

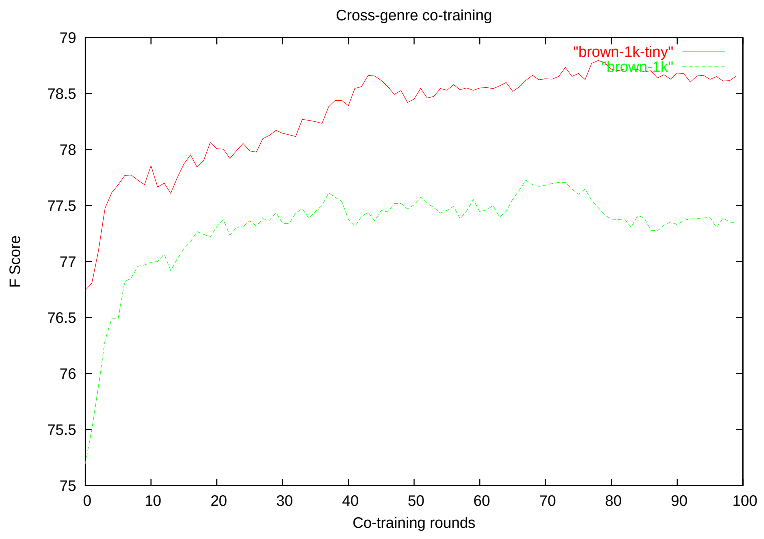


Figure 15: Cross-genre experiments using 1000 sentences seed labeled data from the Brown corpus and co-training and testing on sentences from the Wall Street Journal (evaluation shown for the Collins-CFG parser)

which a small number of **combinatory rules** are used to combine constituents. For example, one of the categories for the verb *likes* specifies that one noun phrase (*NP*) is required to the right of the verb, and one to the left, resulting in a declarative sentence ($S[dcl]$) (as in *John likes sweets*):

$$\text{likes} := (S[dcl]\backslash NP)/NP$$

Another category for *likes* specifies that a to-infinitival clause is required to the right of the verb (as in *John likes to eat sweets*):

$$\text{likes} := (S[dcl]\backslash NP)/(S[\mathrm{to}]\backslash NP)$$

The following derivation shows how categories combine. This derivation uses only two combinatory rules: forward application ($>$) and backward application ($<$).

$$\begin{array}{ccccc}
\text{John} & \text{likes} & \text{to} & \text{eat} & \text{sweets}\\
\overline{NP} & \overline{(S[dcl]\backslash NP)/(S[to]\backslash NP)} & \overline{(S[to]\backslash NP)/(S[b]\backslash NP)} & \overline{(S[b]\backslash NP)/NP} & \overline{NP}\\
 & & & \multicolumn{2}{c}{\overline{\hspace{8em}}{>}}\\
 & & & \multicolumn{2}{c}{S[b]\backslash NP}\\
 & & \multicolumn{3}{c}{\overline{\hspace{16em}}{>}}\\
 & & \multicolumn{3}{c}{S[to]\backslash NP}\\
 & \multicolumn{4}{c}{\overline{\hspace{24em}}{>}}\\
 & \multicolumn{4}{c}{S[dcl]\backslash NP}\\
\multicolumn{5}{c}{\overline{\hspace{30em}}{<}}\\
\multicolumn{5}{c}{S[dcl]}
\end{array}$$

Further combinatory rules (composition, type-raising and substitution) are described in Steedman 2000, which provides a complete introduction to CCG.

We have previously developed two wide-coverage statistical parsers for CCG (Clark et al. 2002; Hockenmaier and Steedman 2002b). Following work in LTAG (Bangalore and Joshi 1994), we have also developed a supertagger for CCG (Clark 2002), which is a program that assigns lexical categories to words, but does not provide derivations. The parsers and the supertagger are trained on CCGbank (Hockenmaier and Steedman 2002a), a corpus of CCG derivations obtained from the Penn Treebank. As in Sarkar (2001), in our co-training experiments on CCG, the two views were provided by one of the parsers (Hockenmaier and Steedman 2002b) and by the supertagger.

4.1 Co-training within CCG

The co-training experiments described so far have used parsers from different formalisms (CFG and LTAG), exploiting the differences between CFG and LTAG to provide the view independence so important for successful co-training. We considered using the CCG parser together with a CFG or LTAG parser. However, the CCG parser requires CCG derivations as training data, and much of the information required to convert a phrase-structure tree into a CCG derivation, especially trace information, is not present in the output of the other parsers. (See Hockenmaier and Steedman 2002a for the procedure which converts Penn Treebank style phrase-structure trees into CCG derivations.) Hence we decided to apply co-training *within* CCG, following Sarkar 2001.

Sarkar also uses co-training within one grammar formalism (LTAG) to bootstrap a parser. One view is provided by the parser itself, and the other is provided by an LTAG supertagger which assigns elementary trees (or “supertags”) to words Bangalore and Joshi (1994). The supertagger is able to provide an alternative view of an LTAG parse tree because it models the *sequence* of supertags in the tree, rather than the way in which the supertags are combined.

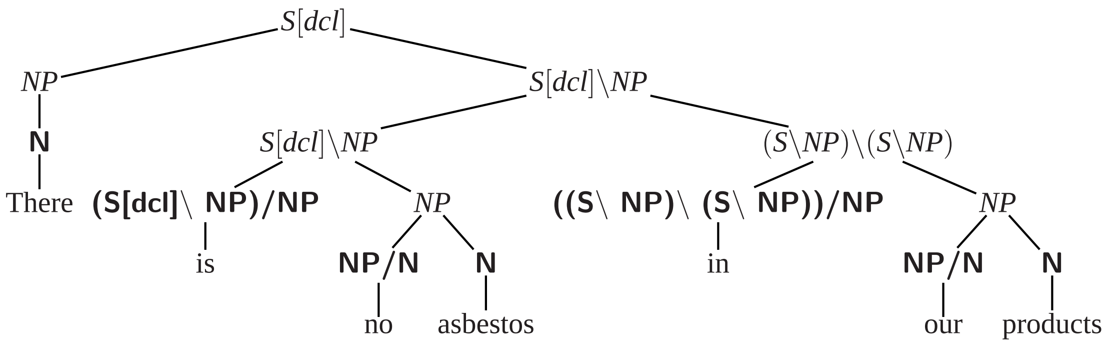


Figure 16: A CCG parse tree with the lexical category sequence in bold

Supertagging can also be applied to CCG and the sequence of lexical categories in a CCG derivation tree can be used to provide a possible view of the tree. Figure 16 shows a (normal-form) CCG derivation tree (with the lexical categories shown in bold). Since lexical categories contain a significant amount of syntactic information (much more than POS-tags), the sequence of categories provides a lot of information about the tree. However, since a supertagger only returns a sequence of lexical categories, the derivation tree itself is still underdetermined. In particular, if there are a number of non-equivalent derivations for the same lexical category sequence the supertagger cannot distinguish between the two. This situation can arise when a modifier (such as a preposition) has multiple possible attachment sites.

The supertagger that we presented originally in Clark 2002 was unsuitable for the co-training experiments because it uses maximum entropy models, and the training algorithm used (generalized iterative scaling) was too slow for retraining on each iteration of the co-training loop. Instead we used an off-the-shelf HMM POS-tagger, namely the TnT tagger (Brants 2000), which performs slightly worse than the maximum entropy supertagger (see Clark 2002), but trains very quickly. TnT can be run in a mode that returns conditional probabilities of tags, and the probabilities for each tag can be multiplied together to give a score for the complete sequence.

The other view of the derivation tree is provided by the parser of Hockenmaier and Steedman 2002b, using the generative probability model employed by the parser. We believed the two views to be complementary because the parser and supertagger are modeling different dependencies in the tree. For example, the supertagger would model a dependency between the lexical categories for *asbestos* and *in* (in the tree in Figure 16), but there is no such dependency between these two categories in the parsing model.

Figure 17 gives the algorithm for how the co-training was performed. The supertagger view is denoted $S+P$ because we needed each view to return complete derivation trees, whereas the supertagger only provides a sequence of categories. Thus for this view we ran the parser in "*n*-best" mode, returning a number of derivation trees for each sentence, and chose a single tree for each sentence on the basis of the probability of the lexical category sequence (obtained by multiplying the individual category probabilities provided by the TnT tagger). In the event of a tie, the probability given by the generative probability model of the parser was used to decide. The *m*-best trees returned by the $S+P$ view were chosen on the basis of the probability of the lexical category sequence, whereas the *m*-best trees returned by the P view were chosen on the basis of the probability of the derivation. In order that the two views did not prefer short trees, both the generative derivation probability and the lexical category sequence probability were normalized by sentence length.

Before presenting the results of the experiments, we show the learning curve for the parser in Figure 18. Here we measure the performance of the parser in terms of the F-score of the constituent-based Parseval measures precision and recall. Since the nonterminal set of CCG is very large and derivation trees are binary branching, these scores are not comparable with Parseval scores for Penn Treebank parsers. We have discussed elsewhere (Clark et al. 2002) alternative evaluation metrics for CCG that are based on word-word dependencies, not constituents.

The next section gives co-training results when a small amount of WSJ Penn Treebank material is used as seed data (1,000 and 10,000 sentences). Section 4.3 gives results for a porting experiment where the seed data is from the Brown section of the Penn Treebank and the unlabled and test data are from the WSJ section. In each case we present three plots: the F-score for the parser, the accuracy of the supertagger, and the accuracy of the parser as a supertagger; that is, how good the parser is at assigning lexical categoies to words. For each experiment the cache size was set at 300 sentences. For the 1,000 seed data experiment, 90 parses were selected at each iteration, and for the other experiments 39 parses were selected at each iteration. These parameters were set on the basis of oracle experiments. The parsing results are based on the first 500 sentences of Section 0, and the supertagger results are based on the complete Section 0.

```
$U$ is a large pool of unlabeled data.
$U'$ is a small cache holding a subset of $U$.
$L_s$ is the set of labeled training examples for the supertagger, $S$.
$L_p$ is the set of labeled training examples for the parser, $P$.
$H_s$ is the current hypothesis of the supertagger.
$H_p$ is the current hypothesis of the parser.

Initialize:
    $L_s = L_p \leftarrow L$.
    $H_s \leftarrow Train(L_s)$
    $H_p \leftarrow Train(L_p)$
Loop
    Move $k$ unlabeled sentences from $U$ to $U'$.
    $S + P$ and $P$ each label the data in $U'$.
    $m$ highest scoring trees according to $P$ are added to $L_s$.
    $m$ highest scoring trees according to $S$ are added to $L_p$.
    $H_s \leftarrow Train(L_s)$
    $H_p \leftarrow Train(L_p)$
```

Figure 17: Pseudo-code for the CCG co-training algorithm

4.2 Using small amounts of WSJ data as the seed

Figure 19 shows how the performance of the parser changes (by F-score) as the co-training proceeds. The x-axis gives the total number of additional sentences added to the 1,000 sentences of seed data. The additional curve shows how the performance changes if the parser is re-trained on its own output (self-training). The performance does improve slightly initially, but then degrades, and co-training performs no better than self-training in this case.

Figure 20 shows how the accuracy of the TnT-supertagger changes as the co-training proceeds. (Note that accuracy here measures the percentage of words which are assigned the correct lexical category.) Here we do see a significant inprovement, from around 81% when trained on the seed data to around 85.5% when $12,000$ new parses have been added. Note that the accuracy of the TnT-

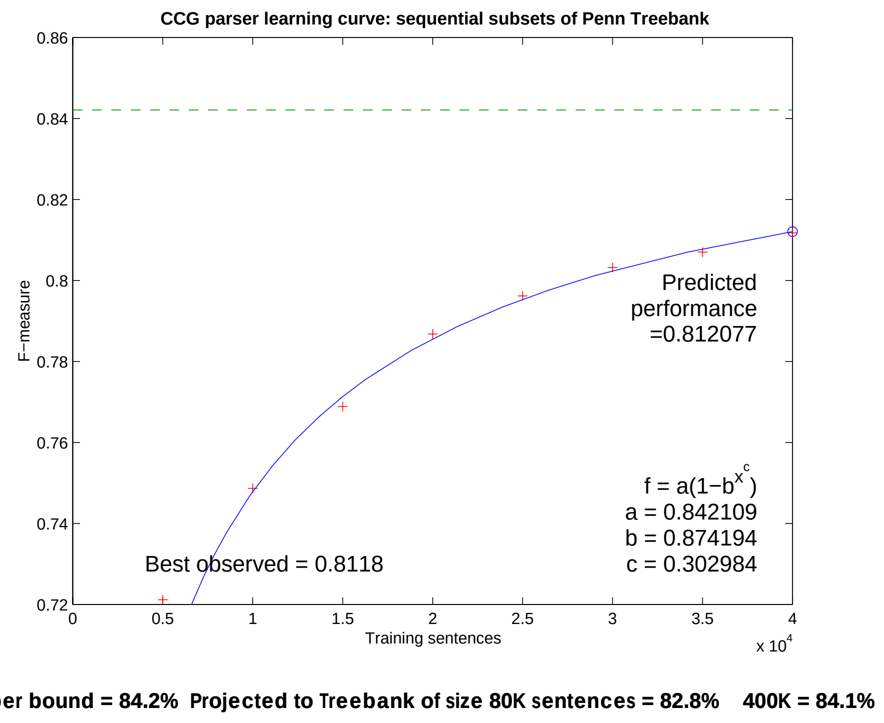


Figure 18: Learning curve for the CCG parser

supertagger is around 89% when trained on the complete CCGbank.[10]

Figure 21 shows how the accuracy of the parser *as a supertagger* changes. Here we do see an improvement, even though the parser as a parser showed no improvement. We also see that co-training performs better than self-training in this case. Interestingly, the parser as a supertagger performs better than the supertagger itself, presumably because the grammar used by the parser provides additional constraints that are not available to the supertagger.

The results for the experiments using 10,000 sentences as seed data show a similar pattern. The performance of the parser shows no significant improvement, but both the supertagger, and the parser as a supertagger, show an improvement. (Figures 22, 23, and 24)

[10]The figure given in Clark 2002 is slightly lower because that figure ignored punctuation.

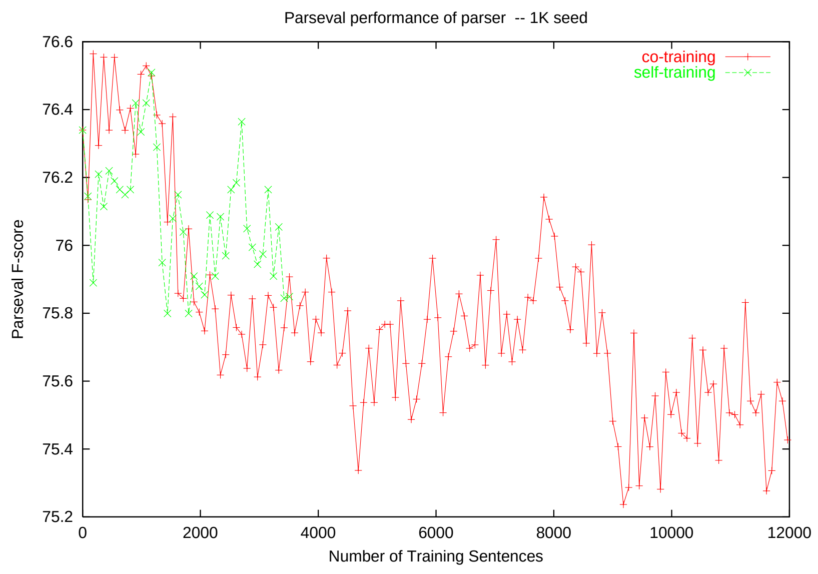


Figure 19: F-score of the parser with 1,000 WSJ sentences as seed data

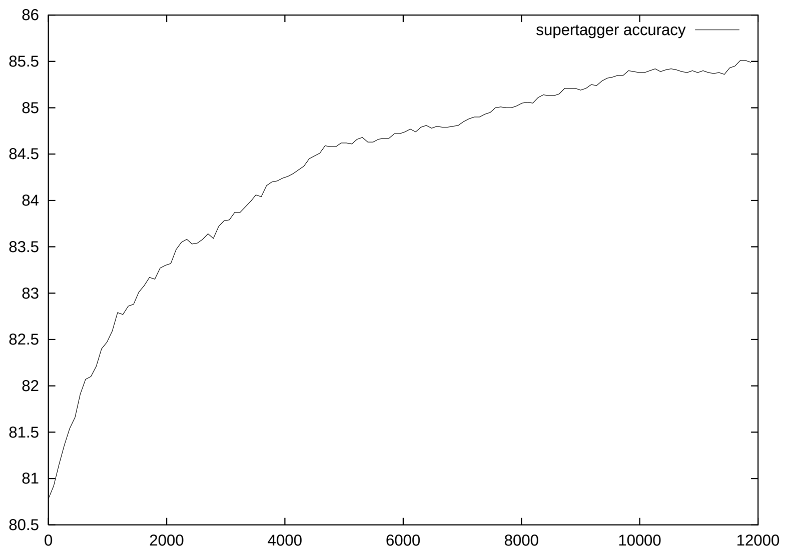


Figure 20: Accuracy of the supertagger with 1,000 WSJ sentences as seed data

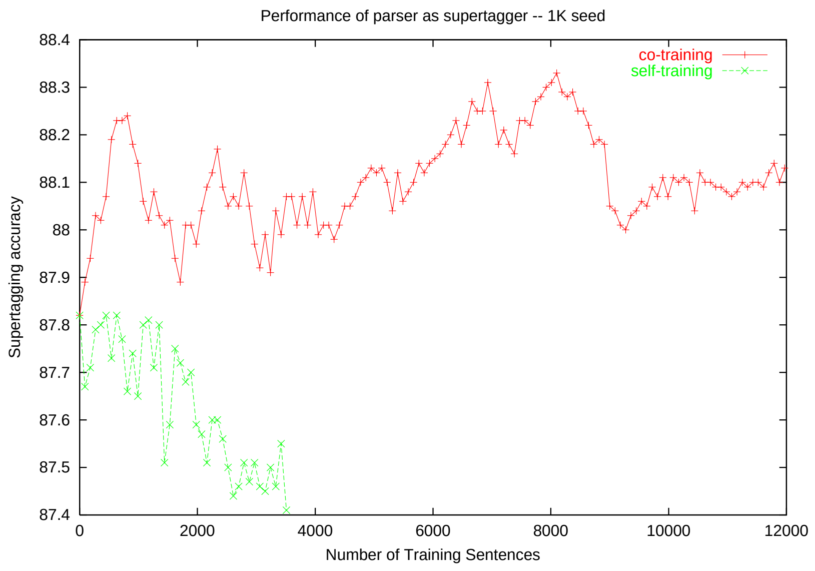


Figure 21: Accuracy of the parser as a supertagger with 1,000 WSJ sentences as seed data

4.3 Porting to another genre: using the Brown corpus as seed

The porting experiment uses the 24,000 sentences from the Brown section of the Penn Treebank as seed data, and then uses the raw sentences from the WSJ section as unlabeled data. Testing is again on Section 0. Here we not only see the supertagger (Figure 26), and the parser as a supertagger (Figure 27), showing an improvement, as before. This time the parser also improves (Figure 25) . Our hypothesis is that the parser improves in this case because the unlabeled data is providing useful information about the new genre which is not available in the seed data.

4.4 Improving the performance of the supertagger trained on the complete Penn Treebank

We investigated whether parses from the Hockenmaier parser could be used to improve the performance of the TnT-supertagger trained on the complete Penn Treebank. The Hockenmaier parser was used to parse a large volume of data from the North American News Text Corpus. By the end of the Workshop around

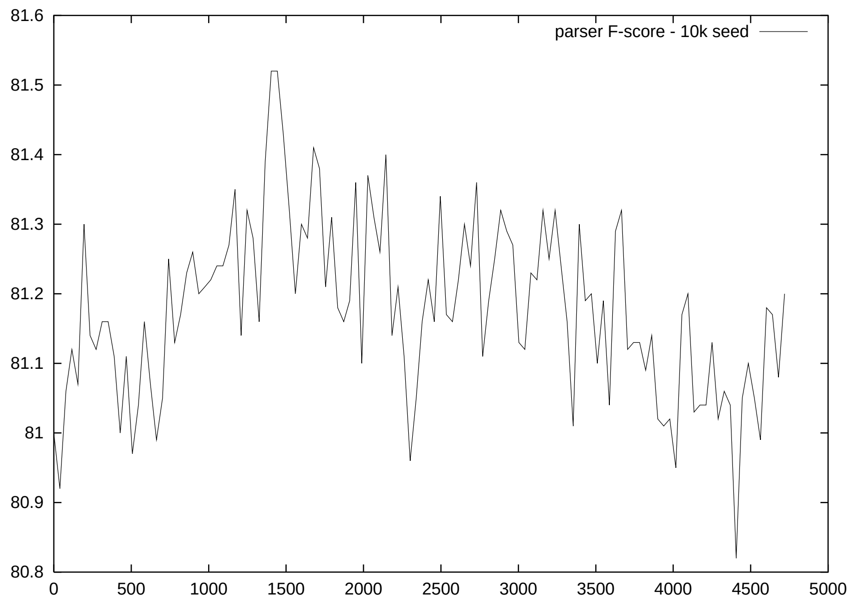


Figure 22: F-score of the parser with 10,000 WSJ sentences as seed data

30 millions words had been parsed. Since the parsing is expensive, it was not possible to do co-training with this much text, but we were able to investigate how the performance of the supertagger changed when various subsets of the parsed material were added to the labeled data.

Figure 28 shows how the accuracy changes as parses are added to the data, where the order in which the parses are added is determined by a number of measures: generative probability, generative probability normalized by sentence length, parse-tree entropy, and parse-tree entropy normalized by sentence length. (See Section 2 for the definition of tree entropy.)

It was hoped that by adding the highest scoring parses first we would be able to use only the high quality parses in the 30 millions words of parsed material. In fact, the plot shows that the scores were unable to distinguish between the good and bad parses since the accuracy continues to increase as the parses with the lowest scores are added to the data. For two of the scores (probability and normalized probability), accuracy even degrades initially when the highest scoring parses are added to the labeled data. However, the large number of parses being added eventually outweighs the noise and the accuracy significantly improves

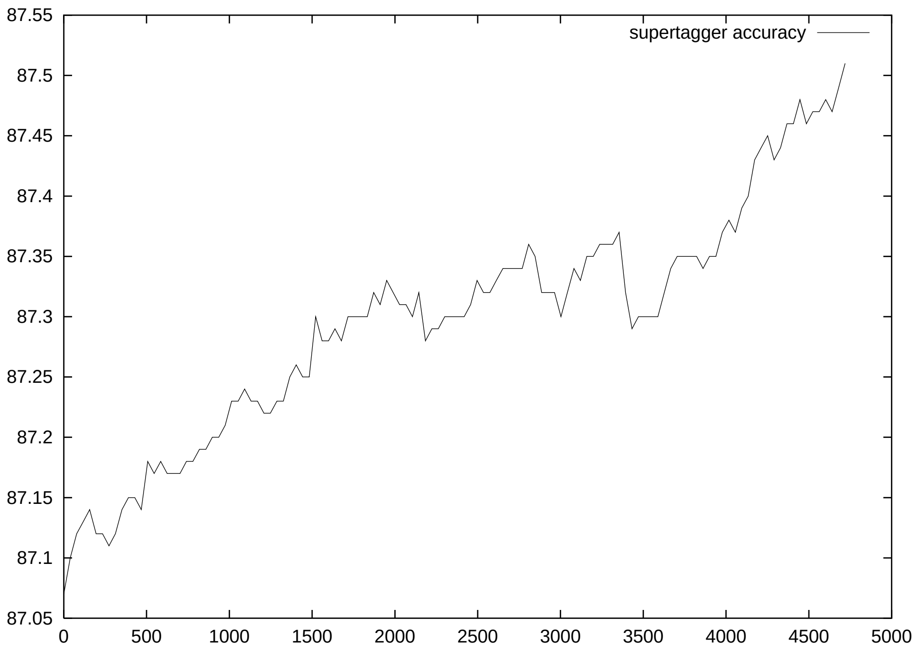


Figure 23: Accuracy of the supertagger with 10,000 WSJ sentences as seed data

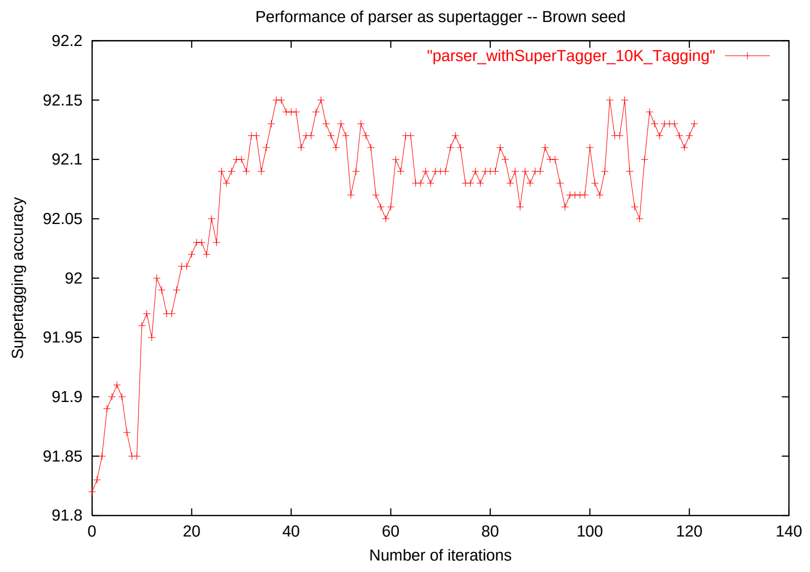


Figure 24: Accuracy of the parser as a supertagger with 10,000 WSJ sentences as seed data

after adding the 30 million words of parsed material.

5 Conclusion

We have investigated co-training of a parser and a supertagger for CCG. We were able to significantly improve the performance of the supertagger under all experimental conditions, and we have also shown that the performance of the parser as a supertagger can be improved through co-training. However, we have not been able to improve the overall performance of the parser by co-training, except in the porting experiment.

If we view CCG parsing as consisting of an initial step of lexical category assignment and of a further step of combining constituents, then what these experiments show is that co-training under the conditions investigated here improves the lexical category assignment of both views. Under our current setup, the attachment decisions that are necessary in determining the derivation are not independent, since both views used the same probability model for them. However, both views differed significantly in the way they assign lexical categories to words, and this is also where we have been able to consistently demonstrate

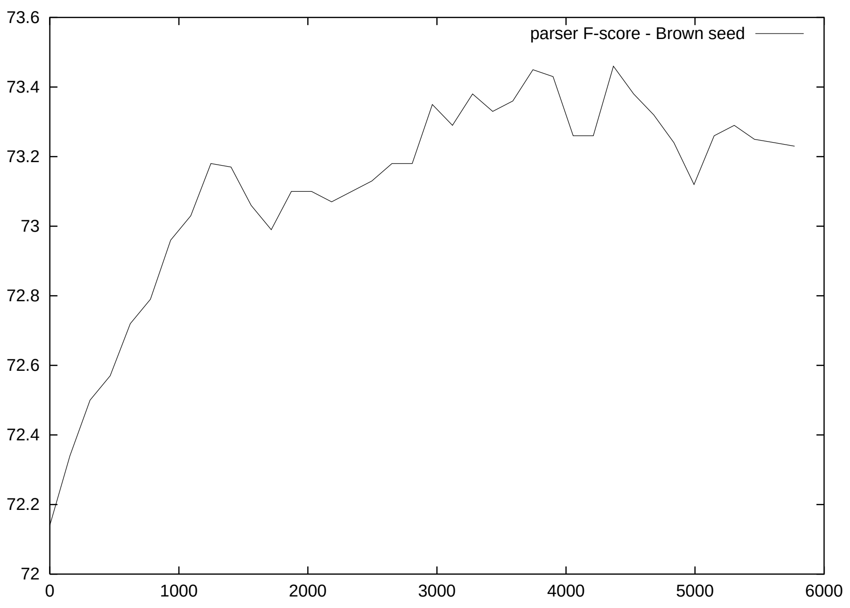


Figure 25: F-score of the parser with 24,000 Brown sentences as seed data

an improvement. In order to fully investigate whether co-training can improve the performance of CCG parsers, a second model of CCG derivations that differs from that in Hockenmaier and Steedman 2002b is required. Such a second view (based on a Maximum Entropy model) is under development by Clark at Edinburgh.

6 Co-training between Parser Re-rankers

The earlier sections have looked at co-training between parsers (and supertaggers). Considering the convergence results of the various parsers, we came to the conclusion that some of the parsers (notably the Collins parser) had already converged (using all of the PTB). This meant that no amount of co-training would improve it. Given this finding, and the fact that the best single model parsing results come from re-ranking the output of a permissive generative model, we looked at co-training between two re-rankers. The general idea is that we only use one parser, which is fixed. However, that parser emits multiple parses for each sentence. These multiple parses are ordered by parser probability. The job of the re-ranker is then to reorder these rankings, with the hope that a better

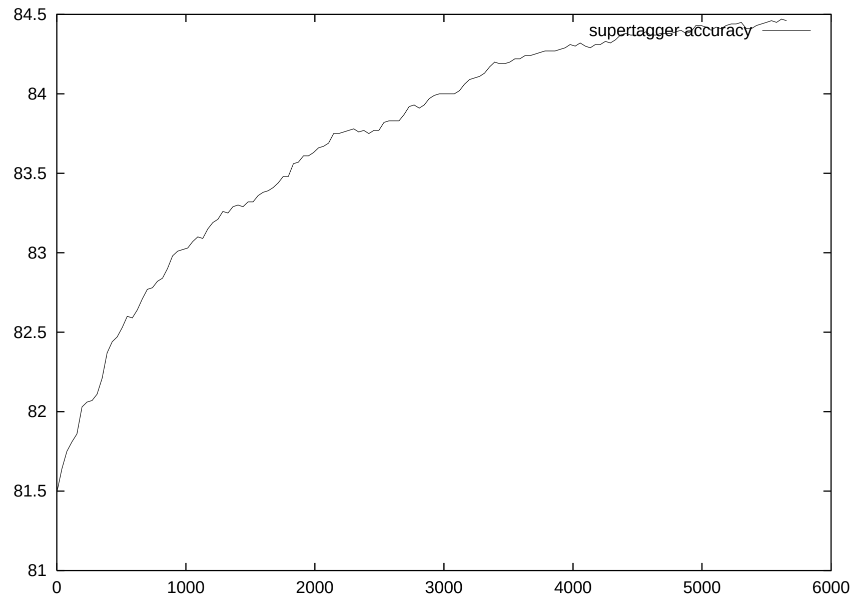


Figure 26: Accuracy of the supertagger with 24,000 Brown sentences as seed data

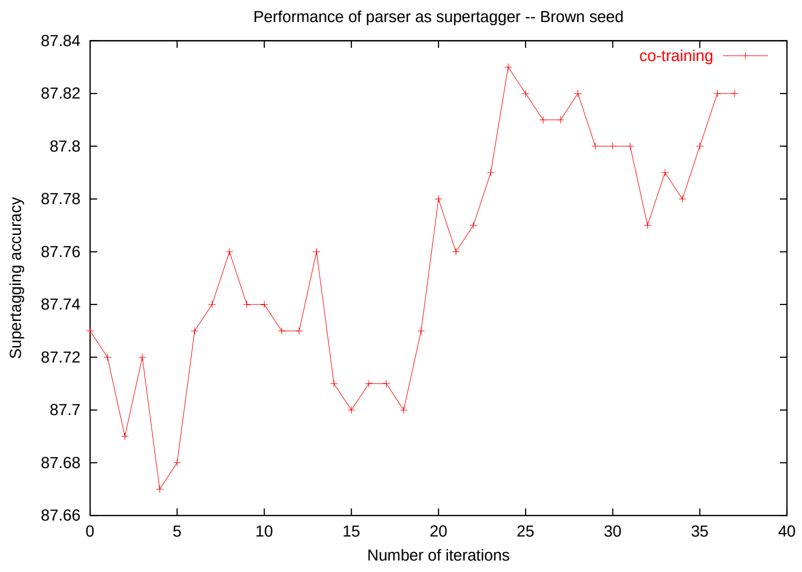


Figure 27: Accuracy of the parser as a supertagger with 24,000 Brown sentences as seed data

parse that was ranked too low down in the list could be promoted to the top of the list. This reordering is possible because the original parser (which is usually generative) usually only considers relatively localized aspects of the tree in its modeling. So, preference decisions which require non-local context cannot be considered. A re-ranker has access to the entire parse, and so can reconsider the rank of a parse, based upon all the evidence. Quite apart from better modeling, from a co-training perspective, re-ranking is attractive since it can be significantly faster than parsing. Also, because the same parser is used, it means that there are no problems when comparing parses with each other. Finally, because re-rankers usually consider a parse as a vector of features, it becomes possible to think about subconstituent co-training. On the negative side, because the same parser is re-ranked, it is possible that our views are less independent than they might be. Also, the general paradigm–according to which a permissive generative model creates n-best parses, and a discriminative re-ranker changes the ordering—does not guarantee that the true positive is within the n-best set.

For our approach, we used the Collins parser as the base model. The two re-rankers were a linear perceptron and a log-linear (maximum entropy) approach.

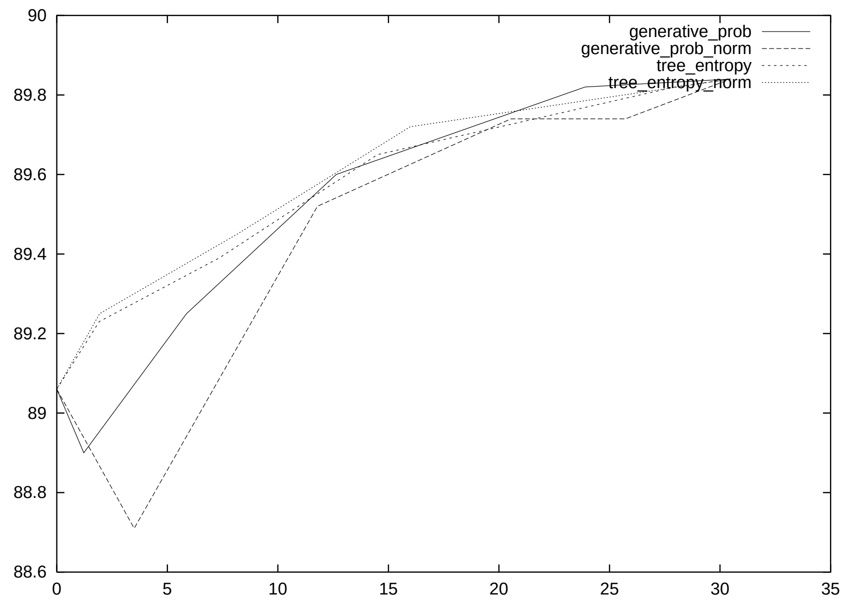


Figure 28: Accuracy of the supertagger with 40,000 WSJ sentences as seed data

Both re-rankers used the same set of features (described later). We also briefly looked at the voted perceptron, winnow and naive Bayes. Due to time constraints, we did not explore these other re-ranking methods for co-training. For parse selection, we found that the voted perceptron and winnow did not outperform the ordinary perceptron.

6.1 Linear perceptron

Following Collins and Duffy, let $h(x)$ correspond to a feature vector representing the parse x. Each sentence in the training set, s_i, has a correct parse t_i. Each sentence also has a list of parses $x_{i,1} \ldots x_{i,n}$. Let this set of parses be the ordering of parses, as produced by the initial parser. Within the set of parses, there will be one or more parses that are ‘closest’ to t_i. Here, closeness is the F-score between the parse and the target parse. Note that this differs from Collins and Duffy in that they assume that the closeness (tree similarity) is 1 for the correct parse, and zero otherwise. Let w be a vector of weights, with one weight per feature. The predicted score of a parse under the perceptron is the dot product of the weights of that parse with the set of weights w $(F(x))$. A set of parses for some sentence can be ranked using $F(x)$.

The perceptron algorithm initializes the weights w to all zeroes. Then, it repeatedly passes over the training set and updates the weights. If some parse x_i has an f-score of j, then if the predicted score of a parse is not j, w is updated as follows:

> Compute the error between the predicted f-score and actual f-score, then subtract this error times the 'learning rate' from the weights of each feature to get the new weights.

For unlabeled parses (parses for which we do not know what the correct parse t_i happens to be), we predicted the rank of a parse as the sum of the weights on that parse. If a feature hadn't been seen before, it would have a weight of zero and so would have no effect. Since the initial rank features are always present, if nothing else in the parse had ever been seen before, it would be assigned the weight of the initial rank feature. For a set of sentences with features that had never been seen before, this would cause the re-ranker to keep the initial ranks for the parses (as produced by the original parser).

6.2 Log-linear modeling

Our log-linear model was similar to the perceptron. In brief, the probability of a parse x given a sentence s_i is:

$$P(x \mid s_i) = \frac{1}{Z(s_i)} \exp(\sum_j \lambda_j f_j) \tag{1}$$

Here, λ_j is the weight corresponding to the feature f_j. These weights are estimated using a limited memory variable metric method.

For unlabeled parses (parses for which we do not know what the correct parse t_i happens to be), we sorted the parses by the sum of the weights of the active features ($\sum_j \lambda_j f_j$). The actual parses were then assigned f-scores as the average F-score that a parse, of that given rank, would have in section 0.

The main differences between the two models are that the log-linear model defines a conditional model (the probability of a parse given a sentence), whereas the perceptron is unconditional. This means that the log-linear model normalizes over the parses for a sentence, irrespective of how good they actually are. The perceptron predicts the actual F-score of a parse, and so better models absolute parse quality. The log-linear model is capable of predicting phenomena which is not linearly separable (it does not make independence assumptions about the features). The perceptron, in contrast, assumes that the data is linearly separable.

6.3 Features

Both models used the same set of features. These were as follows:

- Ngram-like objects. The parse (with non-terminals decorated with percolated lexical items) was walked in a canonical manner producing a sequence of lexicalized non-terminals. Let N_i be the the ith non-terminal, and l_i be the lexicalized item associated with that non-terminal. This sequence was then decomposed into the following ngram-like subsequences:
  - Two contiguous non-terminals N_i and N_{i+1}.
  - N_i and l_I.
  - N_i, l_i, N_{i+1}, l_{i+1}, N_{i+2} and l_{i+2}.
  - N_i, l_i, N_{i+1}, N_{i+2}, N_{i+3}, N_{i+4}, N_{i+5} and N_{i+6}.
- If N_i and N_{i+1} were the same category, then we used a feature that counted the number of such pairings. This feature was designed to model coordination.
- If N_{i+1} was a PP category, then we used the feature N_i, l_{i+1} and l_{i+2}. This feature was an attempt to model some PP attachment decisions.
- We also used a feature which captured some of the modeling information provided by the base parser. Each parse had a rank (as produced by the base CFG parser). We used a rank feature which encoded the rank of the parse. This rank feature introduces distributional information from the base parser into our model.
- Finally, we also used features which corresponded to linguistically plausible units. We used features which counted the number of times a given local unlexicalized tree, of depth one, was seen in a parse.

The motivation for this set of features was speed: we wanted to use features which were cheap to extract from parse trees. Clearly other feature sets are possible.

We found it useful to increase (‘sharpen’) the F-score of the best parse in each sentence in the (seed) training set. This helps differentiate the best parse from the other parses.

Performance using this set of features is state-of-the-art. When using all PTB material, and re-ranking the output of the Collins parser (essentially the same conditions as Collins 00), we achieved an F-score of 89.0 on section 0 and 89.7 using a log-linear model. On section 0, the perceptron achieved an F-score of 89.0. There were 2.4 million features in the model.

6.4 Co-training approach

The previous experiments on co-training parsers selected newly labeled items without any search. Here, we selected new training examples on the basis of explicitly forcing agreement between the two views.

The basic idea was to divide the newly parsed sentences into blocks of sentences and then re-rank each sentence in each block using both views. This produces two sets of blocks of re-ranked parses. Next, for each view, in turn, we took a block of re-reranked parses produced by the *other* re-ranker and re-trained upon that block (in addition to all previously seen re-ranked sentences). We then used the retrained view to re-rank all parses in all blocks, and measured the rank correlation between these rankings and the rankings produced by the other re-ranker. This process was repeated for each block in turn. The block that produced the closest correlation between the rankings of both views was added to the training pool of the re-ranker in question. As can be seen, examples are selected with a view towards (greedily) forcing agreement between the two views.

6.5 Results

We trained the Collins parser on the first 1000 sentences of the PTB and then parsed the remaining training section of the treebank. We processed the newly parsed material into caches of 2500 sentences. Each cache was divided into 5 equal sized blocks. Re-ranking similarity was in terms of Spearman’s rank correlation.

On section 0, prior to re-ranking, the performance of the Collins parser was an F-score of 82.3. The log-linear model’s initial performance (when initialized with 1000 sentences) was an F-score of 82. The linear perceptron (initialized in the same manner) produced an F-score of 82.4.

If the log-linear model was re-trained using the best parses for 10,000 newly parsed sentences, it would achieve an F-score of 83.6. For the perceptron, this score would be 82.8.

After co-training (using how 10,000 sentences in total), the log-linear model slightly improved to an Fscore of 82.4. The perceptron did not improve.

7 General Discussion and Conclusions

The experiments described above have demonstrated the following results.

The experiments show that co-training can enhance performance for parsers and taggers trained on small (500-10,000 sentences) amounts of labeled data—that is, for labeled datasets of the kind of size that can realistically be expected to be obtainable at short notice for novel languages and novel genres of text.

The experiments also show that co-training can be used for porting parsers trained on one genre to parse on another without any new human-labeled data at all. They show in addition that even tiny amounts of human-labeled data for the target genre enhance co-training for porting purposes.

In achieving effective co-training, distinguishing reliable and informative newly labeled data from less reliable and informative output is crucial. Among the most important results of the research are a number of novel methods for parse selection.

The experiments have also yielded some preliminary results on ways to deliver similar improvements for parsers trained on large (Penn WSJ Treebank) labeled datasets and expressive grammars such as TAG and CCG.

These results and some subsequent work by the team are reported in Clark et al. 2003; Baldridge and Osborne 2003; Callison-Burch and Osborne 2003; Osborne et al. 2003; Sarkar et al. 2003; Steedman et al. 2003a,b (see Appendix B: Publications Arising From the Workshop Project). Within the scope of this six-week project, we were not able to show that co-training, given sufficient time and unlimited unlabeled data, can approach the levels of performance attainable with larger amounts of human-labeled data, much less improve upon the state of the art performance of parsers trained on the largest human-labeled datasets. The full scope and limits of parser improvement that can be attained via co-training are a subject for further research.

This is the largest experimental study to date on the use of unlabeled data for improving parser performance. The core result showing that the co-training technique can apply to parsers as well as the classifiers it was originally developed for has wide implications for natural language processing systems.

In the short term, the study shows that in situations where there is an acute shortage of labeled data—because of the novelty of the language or genre involved and the sheer slowness and expense of obtaining human-labeled data—then co-training can be used to enhance performance for a number of types and levels of parser.

In the longer term, as better and less overgenerating parsers are developed, and as our understanding of co-training parsers develops, particularly in terms of better selection methods and levels of co-training below that of whole sentences, the potential payoffs are considerable. Among these are: improved performance for state-of-the-art wide coverage parsers trained on human annotated data; methods for building very large treebanks for training in speech and text-processing applications; improved ease of porting trained parsers to new genres and domains.

References

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

Baldridge, J. and Osborne, M. (2003). Active learning for hpsg parse selection. In Daelemans, W. and Osborne, M., editors, *Computational Natural Language Learning (CoNLL-03)*, Edmonton, Canada. to appear.

Bangalore, S. and Joshi, A. (1994). Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the 15th COLING Conference*, pages 154–160, Kyoto, Japan.

Blum, A. and Mitchell, J. (1998). Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann.

Brants, T. (2000). TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, WA.

Callison-Burch, C. and Osborne, M. (2003). Bootstrapping parallel corpora. In *Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, Edmonton, Canada. to appear.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference of the American Association for Artificial Intelligence, Providence, RI., July*, pages 598–603.

Clark, S. (2002). A supertagger for combinatory categorial grammar. In *Proceedings of the TAG+ Workshop*, pages 19–24, Venice.

Clark, S., Curran, J., and Osborne, M. (2003). Bootstrapping pos-taggers using unlabelled data. In Daelemans, W. and Osborne, M., editors, *Computational Natural Language Learning (CoNLL-03)*, Edmonton, Canada. to appear.

Clark, S., Hockenmaier, J., and Steedman, M. (2002). Building deep dependency structures with a wide-coverage ccg parser. In *Proceedings of the 40th Meeting of the ACL*, pages 327–334, Philadelphia, PA.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Collins, M. (2001). Discriminative reranking for natural language parsing. In *Proceedings of the International Conference on Machine Learning,*.

Dasgupta, S., Littman, M. L., and McAllester, D. (2002). Pac generalization bounds for co-training. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Henderson, J. C. and Brill, E. (1999). Exploiting diversity for natural language processing: Combining parsers. In *Proceedings of the 1999 Joint SIGDAT Conference on Joint Sigdat Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Hockenmaier, J. and Steedman, M. (2002a). Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third LREC Conference*, pages 1974–1981, Las Palmas, Spain.

Hockenmaier, J. and Steedman, M. (2002b). Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia, PA.

Hwa, R. (2000). Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint SIGDAT Conference on EMNLP and VLC*, pages 45–52, Hong Kong, China.

Magerman, D. (1994). *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University University.

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

Nigam, K. and Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proc. of Ninth International Conference on Information and Knowledge (CIKM)*, pages 86–93.

Osborne, M., Hwa, R., and Sarkar, A. (2003). Grammar Induction using Co-training. In preparation.

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

Sarkar, A. (2001). Applying co-training methods to statistical parsing. In *Proceedings of NAACL 2001. Pittsburgh, PA, June*. Morgan Kaufmann.

Sarkar, A. (2002). *Statistical Parsing Algorithms for Lexicalized Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania.

Sarkar, A., Hwa, R., Osborne, M., and Steedman, M. (2003). Corrected-co-training for statistical parsing. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*. In preparation.

Steedman, M. (2000). *The Syntactic Process*. The MIT Press, Cambridge, MA.

Steedman, M., Hwa, R., Clark, S., Osborne, M., Sarkar, A., Hockenmaier, J., Ruhlen, P., Baker, S., and Crim, J. (2003a). Example selection for bootstrapping statistical parsers. In *The Proceedings of the Joint Conference of Human Language Technologies and the Annual Meeting of the North American Chapter of the ACL*.

Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., and Crim, J. (2003b). Bootstrapping statistical parsers from small datasets. In *The Proceedings of the Annual Meeting of the European Chapter of the ACL*.

Yarowsky, D. (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.

Appendix A: Researchers and Addresses

Name	**E-Mail**	**Affiliation**
Researchers:		
Steedman, Mark	steedman@informatics.ed.ac.uk	University of Edinburgh
Sarkar, Anoop	anoop@linc.cis.upenn.edu	University of Pennsylvania
Osborne, Miles	osborne@cogsci.ed.ac.uk	University of Edinburgh
Hwa, Rebecca	hwa@umiacs.umd.edu	University of Maryland
Clark, Stephen	stephenc@cogsci.ed.ac.uk	University of Edinburgh
Hockenmaier, Julia	julia@cogsci.ed.ac.uk	University of Edinburgh
Ruhlen, Paul	ruhlen@cs.jhu.edu	Johns Hopkins University
Baker, Steven	sdb22@cornell.edu	Cornell University
Crim, Jeremiah	jcrim@jhu.edu	Johns Hopkins University
Associates:		
John Henderson	jhndrsn@mitre.org	Mitre Corp.
Chris Callison-Burch	callison-burch@ed.ac.uk	Stanford/Edinburgh

Appendix B: Publications Arising from the Workshop Project

References

Baldridge, J. and Osborne, M. (2003). Active learning for HPSG parse selection. In Daelemans, W. and Osborne, M., editors, *Computational Natural Language Learning (CoNLL-03)*, Edmonton, Canada. to appear.

Callison-Burch, C. and Osborne, M. (2003). Bootstrapping parallel corpora. In *Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, Edmonton, Canada. to appear.

Clark, S., Curran, J., and Osborne, M. (2003). Bootstrapping POS-taggers using unlabeled data. In Daelemans, W. and Osborne, M., editors, *Computational Natural Language Learning (CoNLL-03)*, Edmonton, Canada. to appear.

Osborne, M., Hwa, R., and Sarkar, A. (2003). Grammar Induction using Co-training. In preparation.

Sarkar, A., Hwa, R., Osborne, M., and Steedman, M. (2003). Corrected-co-training for statistical parsing. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*. In preparation.

Steedman, M., Hwa, R., Clark, S., Osborne, M., Sarkar, A., Hockenmaier, J., Ruhlen, P., Baker, S., and Crim, J. (2003a). Example selection for bootstrapping statistical parsers. In *The Proceedings of the Joint Conference of Human Language Technologies and the Annual Meeting of the North American Chapter of the ACL*. to appear.

Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., and Crim, J. (2003b). Bootstrapping statistical parsers from small datasets. In *The Proceedings of the Annual Meeting of the European Chapter of the ACL*. to appear.