Corrected Co-training for Statistical Parsers

Rebecca Hwa HWA@UMIACS.UMD.EDU
Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA

Miles Osborne MILES@INF.ED.AC.UK
School of Informatics, University of Edinburgh, 2 Buccleuch Place, Edinburgh, EH8 9LW, Scotland, UK

Anoop Sarkar ANOOP@CS.SFU.EDU
School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC V5A 1S6, Canada

Mark Steedman STEEDMAN@INF.ED.AC.UK
School of Informatics, University of Edinburgh, 2 Buccleuch Place, Edinburgh, EH8 9LW, Scotland, UK

Abstract

Corrected co-training (Pierce & Cardie, 2001) and the closely related *co-testing* (Muslea et al., 2000) are active learning methods which exploit redundant views to reduce the cost of manually creating labeled training data. We extend these methods to statistical parsing algorithms for natural language. Because creating complex parse structures by hand is significantly more time-consuming than selecting labels from a small set, it may be easier for the human to *correct* the learner's partially accurate output rather than generate the complex label from scratch. The goal of our work is to minimize the number of corrections that the annotator must make. To reduce the human effort in correcting machine parsed sentences, we propose a novel approach, which we call *one-sided corrected co-training* and show that this method requires only a third as many manual annotation decisions as corrected co-training/co-testing to achieve the same improvement in performance.

1. Introduction

Acquiring sufficient quantities of usefully labeled training examples is a major bottleneck for many supervised learning algorithms. This is especially true for tasks where the output is a complex structure rather than an N-ary atomic classification.

Two promising approaches to address the annotation bottleneck are *sample selection*, a variant of active learning (Cohn et al., 1994), in which the learner finds training examples that are the most informative for the human to label, and *co-training* (Blum & Mitchell, 1998), in which two (or more) learners label training examples for each other. More recently, researchers have begun to explore ways of combining ideas from sample selection with that of co-training. One example of this approach is *corrected co-training* as proposed by Pierce and Cardie (2001). This approach inserts a person into the co-training framework so that the machine labeled examples are reviewed and corrected by the human before being added to the training set. A related work in a similar spirit is *co-testing* (Muslea et al., 2000). Like co-training, the framework consists of two classifiers with redundant views, but instead of using the learners' outputs as new training examples, the algorithm uses them as a test for finding informative examples for a human to label. More specifically, the algorithm compares the outputs produced by the learners for the same example. If they disagree, then the example is considered a *contention point*, and therefore a good candidate for human labeling.

In practice, co-testing and corrected co-training have the same effect (i.e., a human is asked to arbitrate over examples selected by co-training); however, they have a subtly different philosophy. The former emphasizes sample selection while the latter emphasizes co-training.

In this work, we extend these combined sample selection and co-training methods to a learning task in which the learner's output is a complex structure. An instance of such a learning problem is training a statistical parser to produce parse trees. Current state-of-the-art statistical parsers (Charniak, 2000; Collins, 1999) are trained on large collections of human parsed sentences such as the Penn Treebank

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

(Marcus et al., 1993). Because producing large-scale treebanks is time-consuming and expensive, there is is considerable interest in methods for reducing the logistic cost of human annotation of training data.

Considering complex structures complicates the problem in two ways. First, because the learner's output is a complex structure, greater nuance is required in accuracy judgment. For example, even if a syntactic tree produced by a parser is not entirely correct, it may have many locally correct parts (such as noun phrase analyses). Thus, in finding contention points, one must determine *how much* of the learners' outputs disagree, and which learner is *more right*. Second, because labeling complex structures is a significantly more time-consuming process, it may be easier for a human to *correct* the learner's partially accurate output rather than generate the label from scratch. Therefore, another major challenge is in finding informative training examples that also require minimal corrections. The complex structure learning task highlights the philosophical differences between co-testing and corrected co-training. To emphasize the corrective nature of the human's task, we shall use the term *corrected co-training* throughout the rest of the paper.

To address these new challenges, we present a method of combining co-training, sample selection and manual intervention for training statistical English parsers called *one-sided corrected co-training*. One-sided co-training consists of two models, A and B, which are used to find a set of contention points. Those points labeled by A are manually corrected before being passed to B. B however, passes to A the examples that it automatically labels. We conduct an empirical study comparing a continuum of approaches, from a fully automatic method (pure co-training), to partially automated methods (one-sided corrected co-training) to fully manual methods (such as corrected co-training and the more traditional single-learner sample selection (Lewis & Catlett, 1994)). It is our hypothesis that partially automated methods such as one-sided corrected co-training can combine the advantages of manual corrections of the output of one parser with the bootstrapping method of co-training and can transfer the benefits of the corrections from one parser to the other thus reducing the effort for the human annotator. Our results suggest that one-sided corrected co-training is an effective way to balance the trade-offs between the performance of the trained parsers, the number of training examples reviewed by a human, and the amount of human effort in making corrections.

2. A Continuum of Bootstrapping Methods

Both sample selection and co-training are iterative learning algorithms that are initialized with a small set of labeled seed data, and both make use of a large pool of unlabeled candidates. The main difference is that sample selection continues to include a human in the training process while co-training does not.

Sample selection can be sub-categorized into uncertainty-based sampling (Lewis & Catlett, 1994) and committee-based sampling (Freund et al., 1997). An uncertainty-based algorithm has a single learner and asks the human to label examples about which it is the least certain. A committee-based algorithm consists of multiple learners (but they do not need to have redundant views) and asks the human to label examples for which the learners disagree the most.

Co-training (Blum & Mitchell, 1998) is also a multi-learner algorithm, but the learners must have different *views* of the data. Blum and Mitchell prove that, when the two views are *conditionally independent* given the label, and each view is sufficient for learning the task, co-training can boost an initial weak learner using unlabeled data. Goldman and Zhou (2000) show that, through careful selection of newly labeled examples, co-training can work even when the classifiers' views do not satisfy the independence assumption. In particular, they show that combining different machine learning algorithms (using the same set of features) using hypothesis testing can result in a successful setting for co-training. As explained in section 4.1, our approach builds upon this idea. We use different grammatical frameworks, with different probabilistic models as the basis of our system.

Figure 1 provides a side-by-side pseudo-code comparison between sample selection and co-training (in particular, for training statistical English parsers).[1] In both cases, the learners are initially trained on a small set of labeled sentences, L. At each training iteration, a small set of sentences is drawn from a large pool of unlabeled sentences and stored in a *cache*. The parsers then attempt to label every sentence in the cache. The parsers also assign a confidence score to their outputs using some scoring function f. Next, both algorithms must select a subset of the newly labeled sentences to be added to the training data. In the sample selection case, the n sentences with the lowest confidence scores are selected for a human to label. The process is a little more involved in the co-training case. The examples added to the training set of one parser (referred to as the *student*) are only those produced by the other parser (referred to as the *teacher*).[2] During selection, one parser first acts as the teacher and the other as the student, and then the roles are reversed. The labeled sentences are chosen

[1] The pseudo-code for the single-learner uncertainty-based sample selection algorithm can be generalized to a committee-based algorithm if we redefine M_A^0 to be a set of parsers $M_{A_j}^0$ that have different initial parameters and f to assign high uncertainty scores to sentences for which the parsing models produced the most different parse trees.

[2] The methods we use generalize to the case in which the parsers share a single training set.

according to some *selection method*, S based on the scores assigned by f. Both algorithms terminate when either all of the sentences in the large pool have been labeled, or when a predetermine number of rounds have been reached.

Pierce and Cardie (2001) while investigating the application of co-training for noun-phrase chunking discovered that the output of the bootstrapped classifiers introduced too much noise into the labeled set, thereby reducing their overall performances. They show that *corrected co-training* could reduce the impact of noise.

Muslea et al. (2000) propose a approach called *co-testing* for sample selection which exploits redundant views to reduce cost for labeling data for classifiers. As shown in (Muslea et al., 2000), *co-testing* is a different class of sample selection technique than uncertainty-based sampling or committee-based sampling. In subsequent work, Muslea et al. (2003) exploit unequal (strong and weak) redundant views in a variant of co-testing called *aggressive co-testing*.

To describe *corrected co-training* (*co-testing*) in terms of Figure 1(b), we change the lines that update the training sets, instead augmenting the training sets with manually corrected examples.

The line $L_A^{i+1} \leftarrow L_A^i \cup \{P_B\}$ is replaced with:

$$L_A^{i+1} \leftarrow L_A^i \cup \text{Corrected}(\{P_B\}),$$

and the line $L_B^{i+1} \leftarrow L_B^i \cup \{P_A\}$ is replaced with:

$$L_B^{i+1} \leftarrow L_B^i \cup \text{Corrected}(\{P_A\}).$$

For *one-sided corrected co-training*, the correction step is applied to just one parser. From a co-training-centric view, the single-learner sample selection algorithm as described in Figure 1(a) can be seen as a form of *corrected self-training*.

In summary, in this section we reviewed a continuum of bootstrapping methods for active learning, and placed our own proposal of *one-sided corrected co-training* into this continuum.

3. Selecting Training Examples for Co-training Parsers

In each iteration of the three variants of co-training algorithms we consider, selection is performed in two steps. First, each parser uses some *scoring function*, f, to assess the parses it generated for the sentences in the cache. The cache is called U^i in Figure 1. In our experiments, all parsers use the same scoring function. Second, a *selection method*, S, is used to choose a subset of these labeled sentences (based on the scores assigned by f) to add to the parsers' training data. This set of examples is similar to what Muslea et al. (2000) referred to as *contention points*.

The scoring function attempts to quantify the correctness of the parses produced by each parser. An ideal scoring function would give the true accuracy rates (e.g., F-score, the combined labeled precision and recall rates). In practice, accuracy is approximated by some notion of confidence. For example, one easy-to-compute scoring function measures the conditional probability of the (most likely) parse. If a high probability is assigned, the parser is said to be confident in the label it produced.

In our experiments, we considered two scoring functions: an oracle scoring function $f_{F\text{-}score}$ that returns the F-score of the parse as measured against a gold standard, and a practical scoring function f_{prob} that returns the conditional probability of the parse.[3] The oracle study shows the effects of correction knowing the absolute quality of machine labeled examples; the practical methods shows the effects of correction as actually witnessed (selected by f_{prob}).

Based on the scores assigned by the scoring function, the selection method chooses a subset of the parser labeled sentences that best satisfy some selection criteria. In our previous work, we have explored different selection methods (Steedman et al., 2003a). For the experiments in this paper, we focus on the *difference* method (denoted as S_{diff}). This selection method selects a sentence (as labeled by the teacher parser) if the score of the teacher's parse is greater than the score of the student's parse by some threshold n.

4. Experiments

Experiments were performed to compare corrected co-training / co-testing against one-sided corrected co-training. Additionally, experiments compared these multi-view active learning methods with a single-view method. As a baseline, we compared all of our active learning methods with co-training.

We used the following four learning methods that used different amounts of manually annotated training data:

- Co-training. This method requires the least amount of human involvement as only the initial training set is human labeled.
- Single-learner sample selection. This method requires the human to review and correct the training examples for one parser.
- Single-sided corrected co-training. This method also requires the human to review and correct the training examples for one parser. It is our hypothesis that the combination of manual corrections of the output

[3] A nice property of using conditional probability, $Pr(parse|sentence)$, as the scoring function is that it normalizes for sentence length.

A is a parser.
M_A^i is the model of A at step i.
U is a set of unlabeled candidates.
U^i is a small cache holding a subset of U at step i.
L is a set of manually labeled seed data.
L_A^i are the labeled training examples for A at step i.
Initialize:
$L_A^0 \leftarrow L$.
$M_A^0 \leftarrow Train(A, L_A^0)$.
Loop:
$U^i \leftarrow$ Add unlabeled sentences from U.
M_A^i parses the sentences in U^i and assigns uncertainty scores to them according to a scoring functions f.
Select the n parses $\{P_A\}$ with the highest scores according to f and remove them from the unlabeled pool.
Ask a person to correct $\{P_A\}$.
$L_A^{i+1} \leftarrow L_A^i \cup Corrected(\{P_A\})$.
$M_A^{i+1} \leftarrow Train(A, L_A^{i+1})$

(a)

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

(b)

Figure 1. The pseudo-code for (a) a single-learner sample selection algorithm and (b) a co-training algorithm

of one parser with the bootstrapping method of co-training and can transfer the benefits of the corrections from one parser to the other thus reducing the effort for the human annotator.

- Corrected co-training. This method requires the human to review and correct training examples for both parsers. Although the human will probably review many more sentences than the others, this method may still be useful if it trains much better parsers and if the human does not need to make many corrections.

The algorithms are judged on several factors: the performance of the trained parser on unseen test sentences, the number of corrections a human has to make, the number of sentences a human has to review, and the convergence rate.

We report the results of two sets of experiments. In the first set of experiments, all the learning algorithms use the oracle scoring function, $f_{F\text{-}score}$. This study shows the effects of correction when we know the absolute amount of noise in each newly parsed sentence. In the second set of experiments, all the learning algorithms use a more practical scoring function, f_{prob}, as a confidence estimator. This study show the effects of correction when examples are selected using parse probabilities. All experiments shared the same general setup, as described below.

4.1. Experimental Setup

For the single-learner sample selection case, we use a lexicalized context free grammar parser developed by Collins (1999) as the learner. For the three co-training variants, we use the Collins parser and a lexicalized Tree Adjoining Grammar parser developed by Sarkar (2002). As we have shown in earlier work, these two parsers are sufficiently distinct from each other for bootstrapping to be effective (Steedman et al., 2003b). For our one-sided corrected co-training experiment, we arbitrarily choose to correct the training examples for the Collins parser. That is, the parses produced by the LTAG parser were manually corrected before being passed to the Collins parser. The parses produced by the Collins parser were not corrected. Note that this means that the Collins parser never sees its own corrected data. The LTAG parser is indirectly improved due to the manual corrections.[4]

For all learning algorithms, all parsers were initialized with the same set of manually labeled training examples (seed data). Since the general goal is to minimize human annotated data, the size of the seed data should be small.

[4]Note that the graphs show the performance of the Collins parser output, however as a result of how one-sided corrected co-training is set up, if the LTAG parser output did not improve then many more manual corrections would have to be made on the LTAG parser output. LTAG parser performance is discussed in Section 5.

In this paper we used a seed set size of $1,000$ sentences, taken from section 2 of the Wall Street Journal (WSJ) Penn Treebank. The total pool of unlabeled sentences was the remainder of sections 2-21 (stripped of their annotations), consisting of about 38,000 sentences. The human effort is measured using the expert human annotation taken from the Penn Treebank. The utility of the output of statistical parsers in real annotation, in terms of the amount of time saved in correcting constituents vs. producing entire trees has been explored in various studies (see (Chiang et al., 2001) for an example of how the Chinese Treebank was augmented using this kind of active learning). In our experiments, the cache size is set at 500 sentences.

The parsers were evaluated on unseen test sentences (section 23 of the WSJ corpus). Section 0 was used as a development set for determining parameters. The evaluation metric is the Parseval F-score over labeled constituents: $F\text{-}score = \frac{2 \times LR \times LP}{LR+LP}$, where LP and LR are labeled precision and recall rate, respectively. For the co-training experiments, both parsers were evaluated, but to be comparable to the sample selection experiment, all results reported here are for the Collins parser.[5]

4.2. Comparison of learning algorithms using the oracle scoring function

Figure 2 compares all four learning algorithms in terms of the parser improvement rates. Each curve in this graph charts the improvement in the parser's accuracy in parsing the test sentences (y-axis) as it is trained on more training sentences (x-axis). The curves have different endpoints because the selection methods chose a different number of sentences from the same 38K unlabeled pool. All three sample selection variants performed significantly better than pure co-training (at the cost of requiring human intervention).

In Figure 3, we focus on the three methods that use sample selection; the graph compares them in terms of the amount of human effort. The graph on the left shows the improvement in the parsing performance (y-axis) as the numbers of sentences reviewed by a person increases (x-axis); the graph on the right shows the improvement as the number of constituents manually corrected increases.

As we can see from Figure 3, our approach of *one-sided corrected co-training* provides the same parsing accuracy as *corrected co-training* but at a substantially reduced cost: the number of constituents that need to be checked when correcting the output of only one parser in the one-sided version is about a third as much as the number for correcting both parsers. Compared to the single-view sample selection method, one-sided corrected co-training requires many fewer constituents to be corrected.

[5]Note that we could produce the best set of constituents rather than the single best tree by a weighted merging of the constituents produced by the two parsers. However, we simply use the output of one parser for convenience.

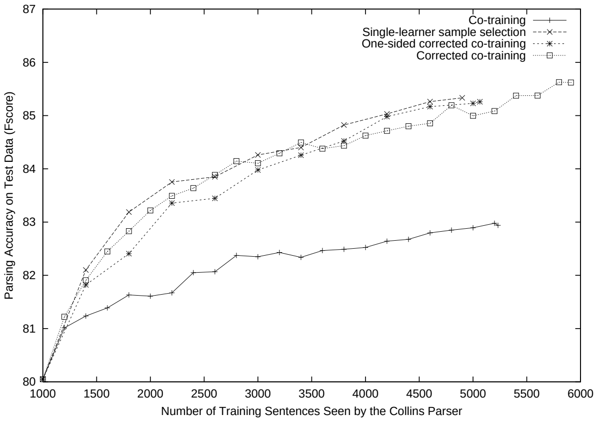


Figure 2. Comparison of the rate of improvements of the learning algorithms when they use an oracle scoring function.

4.3. Comparison of learning algorithms using the parse probability scoring function

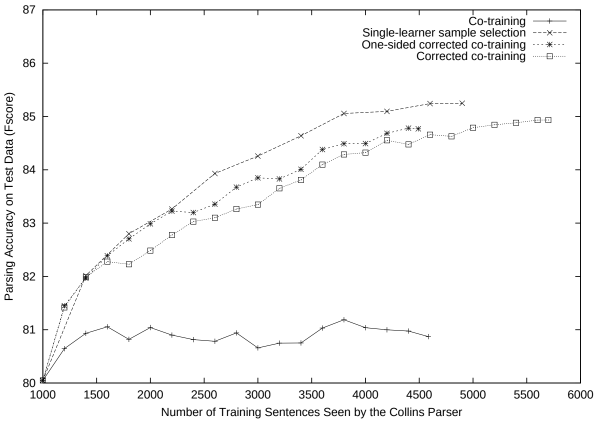
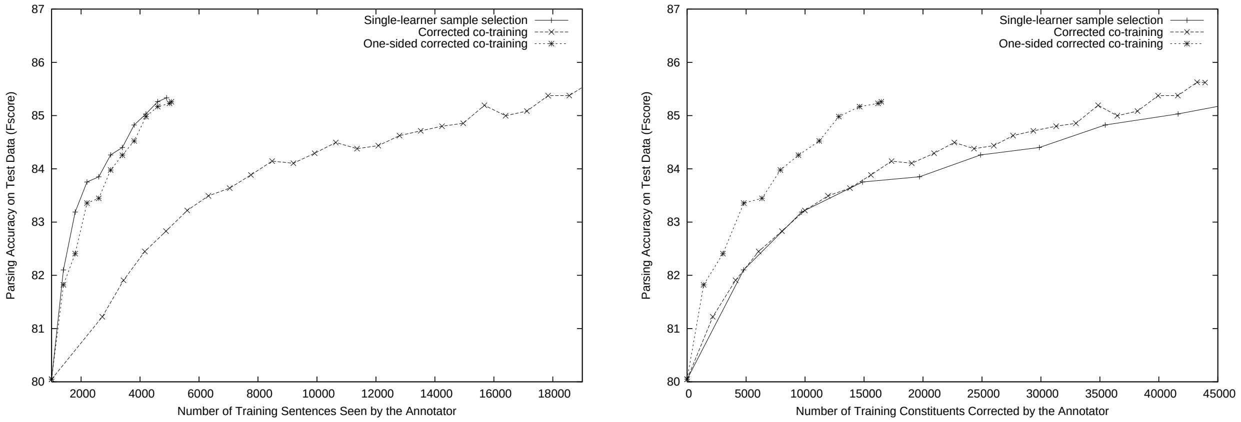


Figure 4. Comparison of the rate of improvements of the learning algorithms when they use the parse probability scoring function.

In the previous set of experiments, we relied on an oracle scoring function to provide the true parsing accuracy rates of the parsers' outputs because it allowed us to study the effects of selections and corrections without complications from whether the parser provides good accuracy estimates. In practice, however, defining a scoring function that gives good accuracy estimates of parser outputs is a difficult problem. In this section, we use the conditional probability of the parse tree as the scoring function. Although very easy to compute (especially when computed from a partially trained parser), the estimates produced will



(a) (b)

Figure 3. Comparison of the three active learning algorithms, using the oracle scoring function to compute their confidence in their outputs. The graph in (a) shows the parsing performances of the parsers in terms of the number of sentences that the human reviewed, and the one in (b) shows parsing performances in terms of the number of constituents that the human corrected.

usually be unreliable.

As in the previous set, we first compare all four learning algorithms for their parser improvement rates in Figure 4. Then, in Figure 5, we compare the three sample selection algorithms in terms of the amount of human effort.

When we retrain our parses with newly labeled examples using a practical scoring function, we see similar findings: both the single-sided corrected co-training and the single view active learning methods require fewer constituents to be manually corrected that does corrected co-training. The same statement can be made regarding the numbers of sentences that need to be reviewed. However, this time we find that one-sided corrected co-training is more profligate in its use of manually labeled examples than before: more examples need to be corrected than before, and difference between this method and the others is less.

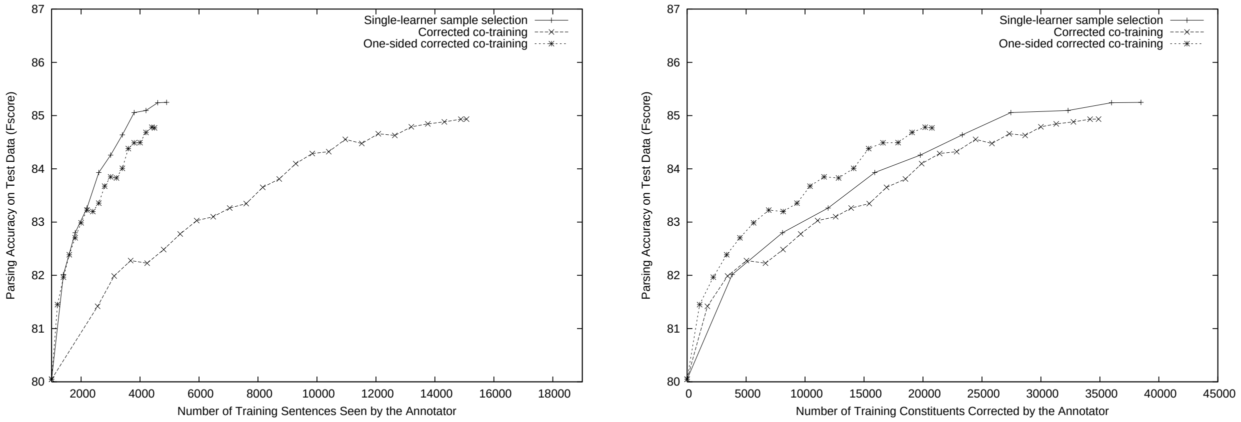
5. Discussion

As our experimental evidence suggests, one-sided corrected co-training is an effective algorithm for finding examples that, when annotated by humans, will lead to more accurate statistical parsers. This shows that the feedback from a second parser helps to substantially reduce the cost of labeling data. Compared with pure co-training, we find that manual correction improves results: some correction is better than none at all. Our study raises two questions.

First, why does *one-sided corrected co-training* do better (i.e., require fewer manual labeling decisions) than *corrected co-training* (*co-testing*)? By definition, correcting the output of one parser is less work than correcting the output of both parsers. But, this is only true if single-sided corrected co-training yields the same level of performance as corrected co-training. It is possible that, within single-sided corrected co-training, the noise levels introduced by the uncorrected view will continue to drag down performance, even though the other view is manually corrected. We find that this is not the case, and that co-training with a mixture of noisily labeled examples (those produced by the uncorrected view) and correctly labeled examples (those that are manually corrected) can be as effective as co-training just using cleanly labeled examples. Single-sided corrected co-training therefore appears to be robust to noise.

Second, will *one-sided corrected co-training* help in cases other than the one we are considering? In cases of classification (where typically there are 2 or 3 classes), labeling agreement is in terms of equality between atomic labels, the particular problem that distinguishes *corrected co-training* from the *one-sided* case does not arise. However, the situation that benefits from *one-sided corrected co-training* is likely to arise in other complex learning problems, for example finding the span and the classification of named-entities in text. Thus, we conjecture that our proposed approach should also benefit the application discussed in Pierce and Cardie (2001).

Note that the performance level of the two parsers are roughly comparable (as shown by the learning curves for the two parsers given in (Steedman et al., 2003a)). However, our experiments in this paper clearly exploit a certain difference between the LTAG parser and the Collins parser: the LTAG parser when trained on less data abstains more often and thus produces fewer trees to correct (explained in detail in (Steedman et al., 2003a)). This means that the use of one-sided corrected co-training leads to fewer man-



(a) (b)

Figure 5. Comparison of the three active learning algorithms, using parse probabilities to estimate the parser's confidence in its outputs. The graph in (a) shows the parsing performances of the parsers in terms of the number of sentences that the human reviewed, and the one in (b) shows parsing performances in terms of the number of constituents that the human corrected.

ual corrections on the output of the LTAG parser as shown in the graphs. The benefit of manual corrections is transferred indirectly via co-training to the Collins parser. We conducted experiments in which the output of the Collins parser was corrected instead (and we tested the Collins parser as before), and this resulted in larger numbers of human corrections with a curve that matches the corrected co-training curves. This is unsurprising, given that co-training is unlikely to match human supervision on the training data.

6. Related Work

Corrected co-training can be seen as a form of active learning, whose goal is to identify the smallest set of unlabeled data with high training utility for the human to label. Active learning can be applied to a single learner (uncertainty sampling) (Lewis & Catlett, 1994) and to multiple learners (committee-based sample selection) (Engelson & Dagan, 1996; Freund et al., 1997; Ngai & Yarowsky, 2000). In the context of parsing, most previous work (Hwa, 2000; Tang et al., 2002; Thompson et al., 1999) has focussed on single learners. Baldridge and Osborne (2003) applied uncertainty sampling and query-by-committee methods for re-ranking parsers. Muslea et al. (2002) introduced the *co-EMT* algorithm for sample selection that is a variation on *co-testing* which combines it with the co-EM algorithm from (Nigam & Ghani, 2000) (a probabilistic variant of co-training). The major difference between co-EM and co-training is that co-EM uses expected fractional counts for the unlabeled examples. However, co-EM is as yet too inefficient to apply to parsing.

7. Future Work

Our current work addresses one particular aspect of our co-training setup: the labeled data created during each co-training iteration is never revisited in subsequent iterations. This was mainly done to improve the processing time needed (statistical parsing is still quite computationally expensive). However, this has a negative result of incorporating noise into labeled data that is never re-evaluated at a later stage (compare our approach with that in (Yarowsky, 1995)). To address this issue, we are running experiments in which each parser, after it is retrained on the manually corrected data, relabels the entire set of previously labeled sentences for the other parser. Thus manual corrections not only improves future labeled data but also possibly corrects mistakes made in previous iterations.

When we corrected trees, we ensured that the entire parse tree was correct. It may be possible that we can further reduce the need for manual correction by only correcting those parts of the parse tree that are crucial for good performance. In addition, we plan to learn from partial parses in future work, taking into account fractional events for parameter updates (borrowing from the EM algorithm) and also by exploiting partial parse features in parse reranking.

8. Conclusion

Applying corrected co-training to an application area such as parsing raises a number of interesting problems, many of which stem from the fact that a parse (which in this case is the output label of the learner) is usually a complicated structure, arising from many interacting decisions. In this paper, we have shown the following:

In terms of parsing performance, all forms of sample selection trained better parsers than pure co-training (at the cost of manual labeling).

Corrected co-training requires fewer corrections than single-learner sample selection. The redundant views used in co-training are shown to be beneficial for the parsing domain.

Corrected co-training, however, requires the human to review many sentences, because outputs from both parsers have to be checked. In contrast, one-sided corrected co-training only asks the human to review and correct sentences for one of the parsers, therefore demands less effort from the human. Moreover, our experimental results show that, despite the inclusion of some errors (from the uncorrected parser labeled training examples), the degradation in parsing performance is small.

Acknowledgments

This work has been supported, in part, by NSF grant 0121285 at JHU, EPSRC grant GR/M96889, the Department of Defense contract RD-02-5700, ONR MURI Contract FCPO.810548265, SFU President's Research Grant and NSERC, Canada RGPIN262313.

References

Baldridge, J., & Osborne, M. (2003). Active learning for HPSG parse selection. *Conference on Computational Natural Language Learning*. ACL.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory* (pp. 92–100). Madison, WI.

Charniak, E. (2000). A maximum-entropy-inspired parser. *Proceedings of the 1st Annual Meeting of the NAACL*. Seattle, Washington.

Chiang, D., Chiou, F.-D., & Palmer, M. (2001). Facilitating treebank annotation using a statistical parser. *Proceedings of HLT 2001, poster session*. San Diego.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*, 201–221.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. Doctoral dissertation, University of Pennsylvania.

Engelson, S. P., & Dagan, I. (1996). Minimizing manual annotation cost in supervised training from copora. *Proceedings of the 34th Annual Meeting of the ACL* (pp. 319–326).

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*, 133–168.

Goldman, S., & Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. *Proceedings of the 17th International Conference on Machine Learning*. Stanford, CA.

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

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 148–156). San Francisco, CA.

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

Muslea, I., Minton, S., & Knoblock, C. (2000). Selective sampling with redundant views. *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (pp. 621–626).

Muslea, I., Minton, S., & Knoblock, C. A. (2002). Active + semi-supervised learning = robust multi-view learning. *ICML-2002*.

Muslea, I., Minton, S., & Knoblock, C. A. (2003). Active learning with strong and weak views: A case study on wrapper induction. *IJCAI-2003*.

Ngai, G., & Yarowsky, D. (2000). Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. *Proceedings of the 38th Annual Meeting of the ACL* (pp. 117–125). Hong Kong, China.

Nigam, K., & Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. *Proceedings of the 9th International Conference on Information and Knowledge Management* (pp. 86–93).

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

Sarkar, A. (2002). *Combining Labeled and Unlabeled Data in Statistical Natural Language Parsing*. Doctoral dissertation, University of Pennsylvania.

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

Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., & Crim, J. (2003b). Bootstrapping statistical parsers from small datasets. *The Proceedings of the Annual Meeting of the European Chapter of the ACL* (pp. 331–338). Budapest, Hungary.

Tang, M., Luo, X., & Roukos, S. (2002). Active learning for statistical natural language parsing. *Proceedings of the 40th Annual Meeting of the ACL* (pp. 120–127). Philadelphia.

Thompson, C. A., Califf, M. E., & Mooney, R. J. (1999). Active learning for natural language parsing and information extraction. *Proceedings of ICML-99* (pp. 406–414). Bled, Slovenia.

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