When is Self-Training Effective for Parsing?

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

Abstract

Self-training has been shown capable of improving on state-of-the-art parser performance (McClosky et al., 2006) despite the conventional wisdom on the matter and several studies to the contrary (Charniak, 1997; Steedman et al., 2003). However, it has remained unclear when and why self-training is helpful. In this paper, we test four hypotheses (namely, presence of a phase transition, impact of search errors, value of non-generative reranker features, and effects of unknown words). From these experiments, we gain a better understanding of why self-training works for parsing. Since improvements from self-training are correlated with unknown bigrams and biheads but not unknown words, the benefit of self-training appears most influenced by seeing known words in new combinations.

1 Introduction

Supervised statistical parsers attempt to capture patterns of syntactic structure from a labeled set of examples for the purpose of annotating new sentences with their structure (Bod, 2003; Charniak and Johnson, 2005; Collins and Koo, 2005; Petrov et al., 2006; Titov and Henderson, 2007). These annotations can be used by various higher-level applications such as semantic role labeling (Pradhan et al., 2007) and machine translation (Yamada and Knight, 2001).

However, labeled training data is expensive to annotate. Given the large amount of unlabeled text available for many domains and languages, techniques which allow us to use both labeled and unlabeled text to train our models are desirable. These methods are called semi-supervised. Self-training is a specific type of semi-supervised learning. In self-training, first we train a model on the labeled data and use that model to label the unlabeled data. From the combination of our original labeled data and the newly labeled data, we train a second model – our self-trained model. The process can be iterated, where the self-trained model is used to label new data in the next iteration. One can think of self-training as a simple case of co-training (Blum and Mitchell, 1998) using a single learner instead of several. Alternatively, one can think of it as one step of the Viterbi EM algorithm.

Studies prior to McClosky et al. (2006) failed to show a benefit to parsing from self-training (Charniak, 1997; Steedman et al., 2003). While the recent success of self-training has demonstrated its merit, it remains unclear why self-training helps in some cases but not others. Our goal is to better understand when and why self-training is beneficial.

In Section 2, we discuss the previous applications of self-training to parsing. Section 3 describes our experimental setup. We present and test four hypotheses of why self-training helps in Section 4 and conclude with discussion and future work in Section 5.

2 Previous Work

To our knowledge, the first reported uses of self-training for parsing are by Charniak (1997). He used his parser trained on the Wall Street Journal (WSJ, Mitch Marcus et al. (1993)) to parse 30 million words of unparsed WSJ text. He then trained

Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 561–568
Manchester, August 2008

a self-trained model from the combination of the newly parsed text with WSJ training data. However, the self-trained model did not improve on the original model.

Self-training and co-training were subsequently investigated in the 2002 CLSP Summer Workshop at Johns Hopkins University (Steedman et al., 2003). They considered several different parameter settings, but in all cases, the number of sentences parsed per iteration of self-training was relatively small (30 sentences). They performed many iterations of self-training. The largest seed size (amount of labeled training data) they used was 10,000 sentences from WSJ, though many experiments used only 500 or 1,000 sentences. They found that under these parameters, self-training did not yield a significant gain.

Reichart and Rappoport (2007) showed that one can self-train with only a generative parser if the seed size is small. The conditions are similar to Steedman et al. (2003), but only one iteration of self-training is performed (i.e. all unlabeled data is labeled at once).[1] In this scenario, unknown words (words seen in the unlabeled data but not in training) were a useful predictor of when self-training improves performance.

McClosky et al. (2006) showed that self-training improves parsing accuracy when the two-stage Charniak and Johnson (2005) reranking parser is used. Using both stages (a generative parser and discriminative reranker) to label the unlabeled data set is necessary to improve performance. Only retraining the first stage had a positive effect. However, after retraining the first stage, both stages produced better parses. Unlike Steedman et al. (2003), the training seed size is large and only one iteration of self-training is performed. Error analysis revealed that most improvement comes from sentences with lengths between 20 and 40 words. Surprisingly, improvements were also correlated with the number of conjunctions but not with the number of unknown words in the sentence.

To summarize, several factors have been identified as good predictors of when self-training improves performance, but a full explanation of why self-training works is lacking. Previous work establishes that parsing all unlabeled sentences at once (rather than over many iterations) is important for successful self-training. The full effect of seed size and the reranker on self-training is not well understood.

[1] Performing multiple iterations presumably fails because the parsing models become increasingly biased. However, this remains untested in the large seed case.

3 Experimental Setup

We use the Charniak and Johnson reranking parser (outlined below), though we expect many of these results to generalize to other generative parsers and discriminative rerankers. Our corpora consist of WSJ for labeled data and NANC (North American News Text Corpus, Graff (1995)) for unlabeled data. We use the standard WSJ division for parsing: sections 2-21 for training (39,382 sentences) and section 24 for development (1,346 sentences). Given self-training's varied performance in the past, many of our experiments use the concatenation of sections 1, 22, and 24 (5,039 sentences) rather than the standard development set for more robust testing.

A full description of the reranking parser can be found in Charniak and Johnson (2005). Briefly put, the reranking parser consists of two stages: A generative lexicalized PCFG parser which proposes a list of the n most probable parses (n-best list) followed by a discriminative reranker which reorders the n-best list. The reranker uses about 1.3 million features to help score the trees, the most important of which is the first stage parser's probability. In Section 4.3, we mention two classes of reranker features in more depth. Since some of experiments rely on details of the first stage parser, we present a summary of the parsing model.

3.1 The Parsing Model

The parsing model assigns a probability to a parse π by a top-down process of considering each constituent c in π and, for each c, first guessing the preterminal of c, $t(c)$ then the lexical head of c, $h(c)$, and then the expansion of c into further constituents $e(c)$. Thus the probability of a parse is given by the equation

$$\begin{aligned} p(\pi) \;=\; & \prod_{c \in \pi} p(t(c) \mid l(c), \mathcal{R}(c)) \\ & \cdot p(h(c) \mid t(c), l(c), \mathcal{R}(c)) \\ & \cdot p(e(c) \mid l(c), t(c), h(c), \mathcal{R}(c)) \end{aligned}$$

where $l(c)$ is the label of c (e.g., whether it is a noun phrase (NP), verb phrase, etc.) and $\mathcal{R}(c)$ is the relevant history of c —- information outside c that the probability model deems important in determining the probability in question.

For each expansion $e(c)$ we distinguish one of the children as the "middle" child $M(c)$. $M(c)$ is the constituent from which the head lexical item h is obtained according to deterministic rules that pick the head of a constituent from among the heads of its children. To the left of M is a sequence of one or more left labels $L_i(c)$ including the special termination symbol $\triangle$ and similarly for the labels to the right, $R_i(c)$. Thus an expansion $e(c)$ looks like:

$$l \rightarrow \triangle L_m ... L_1 M R_1 ... R_n \triangle. \quad (1)$$

The expansion is generated by guessing first M, then in order L_1 through L_{m+1} (= $\triangle$), and similarly for R_1 through R_{n+1}.

So the parser assigns a probability to the parse based upon five probability distributions, T (the part of speech of the head), H (the head), M (the child constituent which includes the head), L (children to the left of M), and R (children to the right of M).

4 Testing the Four Hypotheses

The question of why self-training helps in some cases (McClosky et al., 2006; Reichart and Rappoport, 2007) but not others (Charniak, 1997; Steedman et al., 2003) has inspired various theories. We investigate four of these to better understand when and why self-training helps. At a high level, the hypotheses are (1) self-training helps after a phase transition, (2) self-training reduces search errors, (3) specific classes of reranker features are needed for self-training, and (4) self-training improves because we see new combinations of words.

4.1 Phase Transition

The phase transition hypothesis is that once a parser has achieved a certain threshold of performance, it can label data sufficiently accurately. Once this happens, the labels will be "good enough" for self-training.

To test the phase transition hypothesis, we use the same parser as McClosky et al. (2006) but train on only a fraction of WSJ to see if self-training is still helpful. This is similar to some of the experiments by Reichart and Rappoport (2007) but with the use of a reranker and slightly larger seed sizes. The self-training protocol is the same as in (Charniak, 1997; McClosky et al., 2006; Reichart and Rappoport, 2007): we parse the entire unlabeled corpus in one iteration. We start by taking a random subset of the WSJ training sections (2-21), accepting each sentence with 10% probability. With the sampled training section and the standard development data, we train a parser and a reranker. In Table 1, we show the performance of the parser with and without the reranker. For reference, we show the performance when using the complete training division as well. Unsurprisingly, both metrics drop as we decrease the amount of training data. These scores represent our baselines for this experiment.

Using this parser model, we parse one million sentences from NANC, both with and without the reranker. We combine these one million sentences with the sampled subsets of WSJ training and train new parser models from them.[2]

Finally, we evaluate these self-trained models (Table 2). The numbers in parentheses indicate the change from the corresponding non-self-trained model. As in Reichart and Rappoport (2007), we see large improvements when self-training on a small seed size (10%) without using the reranker. However, using the reranker to parse the self-training and/or evaluation sentences further improves results. From McClosky et al. (2006), we know that when 100% of the training data is used, self-training does not improve performance without a reranker.

From this we conclude that there is no such threshold phase transition in this case. High performance is not a requirement to successfully use self-training for parsing, since there are lower performing parsers which can self-train and higher performing parsers which cannot. The higher performing Charniak and Johnson (2005) parser without reranker achieves an f-score of 89.0 on section 24 when trained on all of WSJ. This parser does not benefit from self-training unless paired with a reranker. Contrast this with the same parser trained on only 10% of WSJ, where it gets an f-score of 85.8 (Table 2) or the small seed models of Reichart and Rappoport (2007). Both of these lower performing parsers can successfully self-train. Additionally, we now know that while a reranker is not required for self-training when the seed size is small, it still helps performance considerably (f-score improves from 87.7 to 89.0 in the 10% case).

[2] We do not weight the original WSJ data, though our expectation is that performance would improve if WSJ were given a higher relative weight. This is left as future work.

% WSJ	# sentences	Parser f-score	Reranking parser f-score
10	3,995	85.8	87.0
100	39,832	89.9	91.5

Table 1: Parser and reranking parser performance on sentences ≤ 100 words in sections 1, 22, and 24 when trained on different amounts of training data. % WSJ is the percentage of WSJ training data trained on (sampled randomly). Note that the full amount of development data is still used as held out data.

Parsed NANC with reranker?	Parser f-score	Reranking parser f-score
No	87.7 (+1.9)	88.7 (+1.7)
Yes	88.4 (+2.6)	89.0 (+2.0)

Table 2: Effect of self-training using only 10% of WSJ as labeled data. The parser model is trained from one million parsed sentences from NANC + WSJ training. The first column indicates whether the million NANC sentences were parsed by the parser or reranking parser. The second and third columns differ in whether the reranker is used to parse the *test* sentences (WSJ sections 1, 22, and 24, sentences 100 words and shorter). Numbers in parentheses are the improvements over the corresponding non-self-trained parser.

4.2 Search Errors

Another possible explanation of self-training's improvements is that seeing newly labeled data results in fewer search errors (Daniel Marcu, personal communication). A search error would indicate that the parsing model could have produced better (more probable) parses if not for heuristics in the search procedure. The additional parse trees may help produce sharper distributions and reduce data sparsity, making the search process easier. To test this, first we present some statistics on the n-best lists ($n = 50$) from the baseline WSJ trained parser and self-trained model[3] from McClosky et al. (2006). We use each model to parse sentences from held-out data (sections 1, 22, and 24) and examine the n-best lists.

We compute statistics of the WSJ and self-trained n-best lists with the goal of understanding how much they intersect and whether there are search errors. On average, the n-best lists overlap by 66.0%. Put another way, this means that about a third of the parses from each model are unique, so the parsers do find a fair number of different parses in their search. The next question is where the differences in the n-best lists lie — if all the differences were near the bottom, this would be less meaningful. Let W and S represent the n-best lists from the baseline WSJ and self-trained parsers, respectively. The $top_m(\ell)$ function returns the highest scoring parse in the n-best list ℓ according to the reranker and parser model m.[4] Almost 40% of the time, the top parse in the self-trained model is not in the WSJ model's n-best list, ($top_s(S) \notin W$) though the two models agree on the top parse roughly 42.4% of the time ($top_s(S) = top_w(W)$). Search errors can be formulated as $top_s(S) \notin W \wedge top_s(S) = top_w(W \cup S)$. This captures sentences where the parse that the reranker chose in the self-trained model is not present in the WSJ model's n-best list, but if the parse were added to the WSJ model's list, the parse's probability in the WSJ model and other reranker features would have caused it to be chosen. These search errors occur in only 2.5% of the n-best lists. At first glance, one might think that this could be enough to account for the differences, since the self-trained model is only several tenths better in f-score. However, we know from McClosky et al. (2006) that on average, parses do not change between the WSJ and self-trained models and when they do, they only improve slightly more than half the time. For this reason, we run a second test more focused on performance.

For our second test we help the WSJ trained model find the parses that the self-trained model found. For each sentence, we start with the n-best list ($n = 500$ here) from the WSJ trained parser, W. We then consider parses in the self-trained parser's n-best list, S, that are not present in W ($S - W$). For each of these parses, we determine its probability under the WSJ trained parsing

[3] http://bllip.cs.brown.edu/selftraining/

[4] Recall that the parser's probability is a reranker feature so the parsing model influences the ranking.

Model	f-score
WSJ	91.5
WSJ & search help	91.7
Self-trained	92.0

Table 3: Test of whether "search help" from the self-trained model impacts the WSJ trained model. WSJ + search help is made by adding self-trained parses not proposed by the WSJ trained parser but to which the parser assigns a positive probability. The WSJ reranker is used in all cases to select the best parse for sections 1, 22, and 24.

model. If the probability is non-zero, we add the parse to the n-best list W, otherwise we ignore the parse. In other words, we find parses that the WSJ trained model could have produced but didn't due to search heuristics. In Table 3, we show the performance of the WSJ trained model, the model with "search help" as described above, and the self-trained model on WSJ sections 1, 22, and 24. The WSJ reranker is used to pick the best parse from each n-best list. WSJ with search help performs slightly better than WSJ alone but does not reach the level of the self-trained model. From these experiments, we conclude that reduced search errors can only explain a small amount of self-training's improvements.

4.3 Non-generative reranker features

We examine the role of specific reranker features by training rerankers using only subsets of the features. Our goal is to determine whether some classes of reranker features benefit self-training more than others. We hypothesize that features which are not easily captured by the generative first-stage parser are the most beneficial for self-training. If we treat the parser and reranking parser as different (but clearly dependent) views, this is a bit like co-training. If the reranker uses features which are captured by the first-stage, the views may be too similar for there to be an improvement.

We consider two classes of features (GEN and EDGE) and their complements (NON-GEN and NON-EDGE).[5] GEN consists of features that are roughly captured by the first-stage generative parser: rule rewrites, head-child dependencies, etc. EDGE features describe items across constituent boundaries. This includes the words and parts of speech of the tokens on the edges between constituents and the labels of these constituents. This represents a specific class of features not captured by the first-stage. These subsets and their sizes are shown in Table 4. For comparison, we also include the results of experiments using the full feature set, as in McClosky et al. (2006), labeled ALL. The GEN features are roughly one third the size of the full feature set.

[5] A small number of features overlap hence these sizes do not add up.

Feature set	# features	f-score
GEN	448,349	90.4
NON-GEN	885,492	91.1
EDGE	601,578	91.0
NON-EDGE	732,263	91.1
ALL	1,333,519	91.3

Table 4: Sizes and performance of reranker feature subsets. Reranking parser f-scores are on all sentences in section 24.

We evaluate the effect of these new reranker models on self-training (Table 4). For each feature set, we do the following: We parse one million NANC sentences with the reranking parser. Combining the parses with WSJ training data, we train a new first-stage model. Using this new first-stage model and the reranker subset, we evaluate on section 24 of WSJ. GEN's performance is weaker while the other three subsets achieve almost the same score as the full feature set. This confirms our hypothesis that when the reranker helps in self-training it is due to features which are not captured by the generative first-stage model.

4.4 Unknown Words

Given the large size of the parsed self-training corpus, it contains an immense number of parsing events which never occur in the training corpus. The most obvious of these events is words — the vocabulary grows from 39,548 to 265,926 words as we transition from the WSJ trained model to the self-trained model. Slightly less obvious is bigrams. There are roughly 330,000 bigrams in WSJ training data and approximately 4.8 million new bigrams in the self-training corpus.

One hypothesis (Mitch Marcus, personal communication) is that the parser is able to learn a lot of new bilexical head-to-head dependencies (bi-heads) from self-training. The reasoning is as follows: Assume the self-training corpus is parsed in a mostly correct manner. If there are not too many

new pairs of words in a sentence, there is a decent chance that we can tag these words correctly and bracket them in a reasonable fashion from context. Thus, using these parses as part of the training data improves parsing because should we see these pairs of words together in the future, we will be more likely to connect them together properly.

We test this hypothesis in two ways. First, we perform an extension of the factor analysis similar to that in McClosky et al. (2006). This is done via a generalized linear regression model intended to determine which features of parse trees can predict when the self-training model will perform better. We consider many of the same features (e.g. bucketed sentence length, number of conjunctions, and number of unknown words) but also consider two new features: unknown bigrams and unknown biheads. Unknown items (words, bigrams, biheads) are calculated by counting the number of items which have never been seen in WSJ training but have been seen in the parsed NANC data. Given these features, we take the f-scores for each sentence when parsed by the WSJ and self-trained models and look at the differences. Our goal is to find out which features, if any, can predict these f-score differences. Specifically, we ask the question of whether seeing more unknown items indicates whether we are more likely to see improvements when self-training.

The effect of unknown items on self-training's relative performance is summarized in Figure 1. For each item, we show the total number of incorrect parse nodes in sentences that contain the item. We also show the change in the number of correct parse nodes in these sentences between the WSJ and self-trained models. A positive change means that performance improved under self-training. In other words, looking at Figure 1a, the greatest performance improvement occurs, perhaps surprisingly, when we have seen no unknown words. As we see more unknown words, the improvement from self-training decreases. This is a pretty clear indication that unknown words are not a good predictor of when self-training improves performance.

A possible objection that one might raise is that using unknown biheads as a regression feature will bias our results if they are counted from gold trees instead of parsed trees. Seeing a bihead in training will cause the otherwise sparse biheads distribution to be extremely peaked around that bihead. If we see the same pair of words in testing, we are likely to connect them in the same fashion. Thus, if we count unknown biheads from gold trees, this feature may explain away other improvements: When gold trees contain a bihead found in our self-training data, we will almost always see an improvement. However, given the similar trends in Figures 1b and 1c, we propose that unknown bigrams can be thought of as a rough approximation of unknown biheads.

f-score		Model
89.8	$*$	WSJ (baseline)
89.8	$*$	WSJ+NANC M
89.9	$*$	WSJ+NANC T
89.9	$*$	WSJ+NANC L
90.0	$*$	WSJ+NANC R
90.0		WSJ+NANC MT
90.1		WSJ+NANC H
90.2		WSJ+NANC LR
90.3		WSJ+NANC LRT
90.4		WSJ+NANC LMRT
90.4		WSJ+NANC LMR
90.5		WSJ+NANC LRH
90.7	$\circledast$	WSJ+NANC LMRH
90.8	$\circledast$	WSJ+NANC (fully self-trained)

Table 5: Performance of the first-stage parser on various combinations of distributions WSJ and WSJ+NANC (self-trained) models on sections 1, 22, and 24. Distributions are L (left expansion), R (right expansion), H (head word), M (head phrasal category), and T (head POS tag). $*$ and $\circledast$ indicate the model is not significantly different from baseline and self-trained model, respectively.

The regression analysis reveals that unknown bigrams and unknown biheads are good predictors of f-score improvements. The significant predictors from McClosky et al. (2006) such as the number of conjunctions or sentence length continue to be helpful whereas unknown words are a weak predictor at best. These results are apparent in Figure 1: as stated before, seeing more unknown words does not correlate with improvements. However, seeing more unknown bigrams and biheads does predict these changes fairly well. When we have seen zero or one new bigrams or biheads, self-training negatively impacts performance. After seeing two or more, we see positive effects until about six to ten after which improvements taper off.

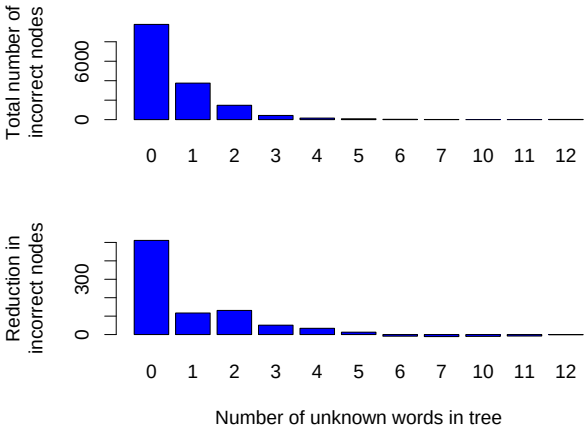
To see the effect of biheads on performance more directly, we also experiment by interpolating between the WSJ and self-trained models on a distribution level. To do this, we take specific distributions (see Section 3.1) from the self-trained model and have them override the corresponding distributions in a compatible WSJ trained model. From this we hope to show which distributions self-training boosts. According to the biheads hypothesis, the H distribution (which captures information about head-to-head dependencies) should account for most of the improvement.

The results of moving these distributions is shown in Table 5. For each new model, we show whether the model's performance is not significantly different than the baseline model (indicated by $*$) or not significantly different than the self-trained model ($\circledast$). H (biheads) is the strongest single feature and the only one to be significantly better than the baseline. Nevertheless, it is only 0.3% higher, accounting for 30% of the full self-training improvement. In general, the performance improvements from distributions are additive ($+/-$ 0.1%). Self-training improves all distributions, so biheads are not the full picture. Nevertheless, they remain the strongest single feature.

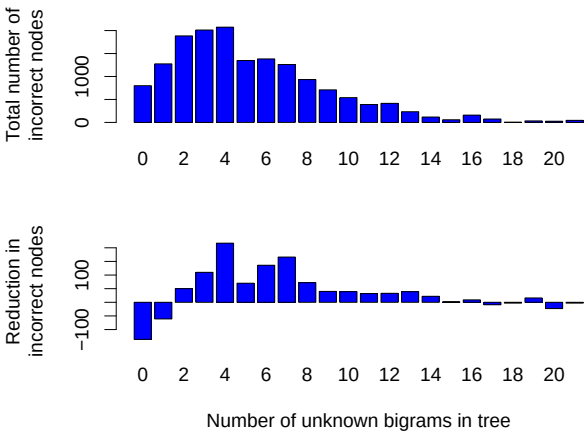
5 Discussion

The experiments in this paper have clarified many details about the nature of self-training for parsing. We have ruled out the phase transition hypothesis entirely. Reduced search errors are responsible for some, but not all, of the improvements in self-training. We have confirmed that non-generative reranker features are more beneficial than generative reranker features since they make the reranking parser more different from the base parser. Finally, we have found that while unknown bigrams and biheads are a significant source of improvement, they are not the sole source of it. Since unknown words do not correlate well with self-training improvements, there must be something about the unknown bigrams and biheads which are aid the parser. Our belief is that new combinations of words we have already seen guides the parser in the right direction. Additionally, these new combinations result in more peaked distributions which decreases the number of search errors.

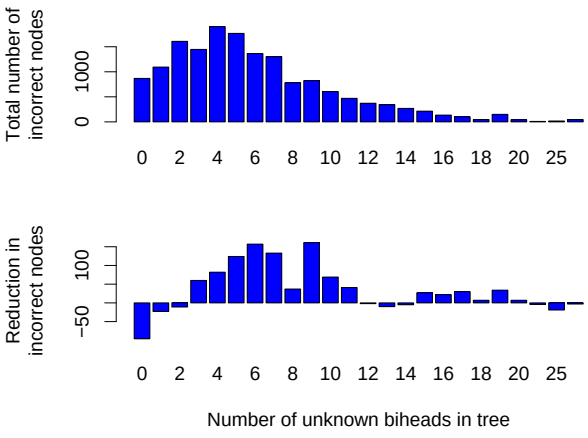
However, while these experiments and others get us closer to understanding self-training, we still lack a complete explanation. Naturally, the hy-



(a) Effect of unknown words on performance



(b) Effect of unknown bigrams on performance



(c) Effect of unknown biheads on performance

Figure 1: Change in the number of incorrect parse tree nodes between WSJ and self-trained models as a function of number of unknown items. Seeing any number of unknown words results in fewer errors on average whereas seeing zero or one unknown bigrams or biheads is likely to hurt performance.

potheses tested are by no means exhaustive. Additionally, we have only considered generative constituency parsers here and a good direction for future research would be to see if self-training generalizes to a broader class of parsers. We suspect that using a generative parser/discriminative reranker paradigm should allow self-training to extend to other parsing formalisms.

Recall that in Reichart and Rappoport (2007) where only a small amount of labeled data was used, the number of unknown words in a sentence was a strong predictor of self-training benefits. When a large amount of labeled data is available, unknown words are no longer correlated with these gains, but unknown bigrams and biheads are. When using a small amount of training data, unknown words are useful since we have not seen very many words yet. As the amount of training data increases, we see fewer new words but the number of new bigrams and biheads remains high. We postulate that this difference may help explain the shift from unknown words to unknown bigrams and biheads. We hope to further investigate the role of these unknown items by seeing how our analyses change under different amounts of labeled data relative to unknown item rates.

Acknowledgments

This work was supported by DARPA GALE contract HR0011-06-2-0001. We would like to thank Matt Lease, the rest of the BLLIP team, and our anonymous reviewers for their comments. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

References

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

Bod, Rens. 2003. An efficient implementation of a new DOP model. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proc. of the 2005 Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 173–180.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. AAAI*, pages 598–603.

Collins, Michael and Terry Koo. 2005. Discriminative Reranking for Natural Language Parsing. *Computational Linguistics*, 31(1):25–69.

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

McClosky, David, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159.

Mitch Marcus et al. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comp. Linguistics*, 19(2):313–330.

Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Pradhan, Sameer, Wayne Ward, and James Martin. 2007. Towards robust semantic role labeling. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 556–563, Rochester, New York, April. Association for Computational Linguistics.

Reichart, Roi and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623.

Steedman, Mark, Steven Baker, Jeremiah Crim, Stephen Clark, Julia Hockenmaier, Rebecca Hwa, Miles Osborne, Paul Ruhlen, and Anoop Sarkar. 2003. CLSP WS-02 Final Report: Semi-Supervised Training for Statistical Parsing. Technical report, Johns Hopkins University.

Titov, Ivan and James Henderson. 2007. Constituent parsing with incremental sigmoid belief networks. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 632–639, Prague, Czech Republic, June. Association for Computational Linguistics.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–529.