Self-Training PCFG Grammars with Latent Annotations Across Languages

Zhongqiang Huang[1]
[1]Laboratory for Computational Linguistics and Information Processing
Institute for Advanced Computer Studies
University of Maryland, College Park
zqhuang@umiacs.umd.edu

Mary Harper[1,2]
[2]Human Language Technology Center of Excellence
Johns Hopkins University
mharper@umiacs.umd.edu

Abstract

We investigate the effectiveness of self-training PCFG grammars with latent annotations (PCFG-LA) for parsing languages with different amounts of labeled training data. Compared to Charniak's lexicalized parser, the PCFG-LA parser was more effectively adapted to a language for which parsing has been less well developed (i.e., Chinese) and benefited more from self-training. We show for the first time that self-training is able to significantly improve the performance of the PCFG-LA parser, a single generative parser, on both small and large amounts of labeled training data. Our approach achieves state-of-the-art parsing accuracies for a single parser on both English (91.5%) and Chinese (85.2%).

1 Introduction

There is an extensive research literature on building high quality parsers for English (Collins, 1999; Charniak, 2000; Charniak and Johnson, 2005; Petrov et al., 2006), however, models for parsing other languages are less well developed. Take Chinese for example; there have been several attempts to develop accurate parsers for Chinese (Bikel and Chiang, 2000; Levy and Manning, 2003; Petrov and Klein, 2007), but the state-of-the-art performance, around 83% F measure on Penn Chinese Treebank (achieved by the Berkeley parser (Petrov and Klein, 2007)) falls far short of performance on English (∼90-92%). As pointed out in (Levy and Manning, 2003), there are many linguistic differences between Chinese and English, as well as structural differences between their corresponding treebanks, and some of these make it a harder task to parse Chinese. Additionally, the fact that the available treebanked Chinese materials are more limited than for English also increases the challenge of building high quality Chinese parsers. Many of these differences would also tend to apply to other less well investigated languages.

In this paper, we focus on English and Chinese because the former is a language for which extensive parsing research has been conducted while the latter is a language that has been less extensively studied. We adapt and improve the Berkeley parser, which learns PCFG grammars with latent annotations, and show through comparative studies that this parser significantly outperforms Charniak's parser, which was initially developed for English and subsequently ported to Chinese. We focus on answering two questions: how well does a parser perform across languages and how much does it benefit from self-training?

The first question is of special interest when choosing a parser that is designed for one language and adapting it to another less studied language. We improve the PCFG-LA parser by adding a language-independent method for handling rare words and adapt it to another language, Chinese, by creating a method to better model Chinese unknown words. Our results show that the PCFG-LA parser performs significantly better than Charniak's parser on Chinese, and is also somewhat more accurate on English, although both parsers have high accuracy.

The second question is important because labeled training data is often quite limited, especially for less well investigated languages, while unlabeled data is ubiquitous. Early investigations on self-training for parsing have had mixed results. Charniak (1997) reported no improvements from self-training a PCFG parser on the standard WSJ training set. Steedman et al. (2003) reported some degradation using a lexicalized tree adjoining grammar parser and minor improvement using Collins lexicalized PCFG parser; however, this gain was obtained only when the parser

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 832–841,
Singapore, 6-7 August 2009.

was trained on a small labeled set. Reichart and Rappoport (2007) obtained significant gains using Collins lexicalized parser with a different self-training protocol, but again they only looked at small labeled sets. McClosky et al. (2006) effectively utilized unlabeled data to improve parsing accuracy on the standard WSJ training set, but they used a two-stage parser comprised of Charniak's lexicalized probabilistic parser with n-best parsing and a discriminative reranking parser (Charniak and Johnson, 2005), and thus it would be better categorized as "co-training" (McClosky et al., 2008). It is worth noting that their attempts at self-training Charniak's lexicalized parser directly resulted in no improvement. There are other successful semi-supervised training approaches for dependency parsing, such as (Koo et al., 2008; Wang et al., 2008), and it would be interesting to investigate how they could be applied to constituency parsing.

We show in this paper, for the first time, that self-training is able to significantly improve the performance of the PCFG-LA parser, a single generative parser, on both small and large amounts of labeled training data, for both English and Chinese. With self-training, a fraction of the WSJ or CTB6 treebank training data is sufficient to train a PCFG-LA parser that is able to achieve or even beat the accuracies obtained using a single parser trained on the entire treebank without self-training. We conjecture based on our comparison of the PCFG-LA parser to Charniak's parser that the addition of self-training data helps the former parser learn more fine-grained latent annotations without over-fitting.

The rest of this paper is organized as follows. We describe the PCFG-LA parser and several enhancements in Section 2, and discuss self-training in Section 3. We then outline the experimental setup in Section 4, describe the results in Section 5, and present a detailed analysis in Section 6. The last section draws conclusions and describes future work.

2 Parsing Model

The Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007) is an efficient and effective parser that introduces latent annotations (Matsuzaki et al., 2005) to refine syntactic categories to learn better PCFG grammars. In the example parse tree in Figure 1(a), each syntactic category is split into multiple latent subcategories, and accordingly the original parse tree is decomposed into many parse trees with latent annotations. Figure 1(b) depicts one of such trees. The grammar and lexical rules are split accordingly, e.g., NP→PRP is split into different NP-i→PRP-j rules. The expansion probabilities of these split rules are the parameters of a PCFG-LA grammar.

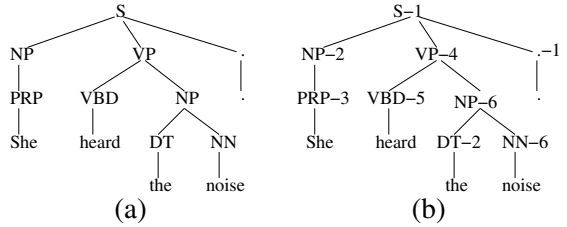


Figure 1: (a) original treebank tree, (b) after latent annotation.

The objective of training is to learn a grammar with latent annotations that maximizes the likelihood of the training trees, i.e., the sum of the likelihood of all parse trees with latent annotations. Since the latent annotations are not available in the treebank, a variant of the EM algorithm is utilized to learn the rule probabilities for them. The Berkeley parser employs a hierarchical split-merge method that gradually increases the number of latent annotations and adaptively allocates them to different treebank categories to best model the training data. In this paper, we call a grammar trained after n split-merge steps an n-th *order* grammar. The order of a grammar is a step (not continuous) function of the number of latent annotations because the split-merge algorithm first splits each latent annotation into two and then merges some of the splits back based on their ability to increase training likelihood.

For this paper, we implemented[1] our own version of Berkeley parser. Updates include better handling of rare words across languages, as well as unknown Chinese words. The parser is able to process difficult sentences robustly using adaptive beam expansion. The training algorithm was updated to support a wide range of self-training experiments (e.g., posterior-weighted unlabeled data, introducing self-training in later iterations) and to make use of multiple processors to parallelize EM training. The parallelization is crucial

[1] A major motivation for this implementation was to support some algorithms we are developing. Most of our enhancements will be merged with a future release of the Berkeley parser.

for training a model with large volumes of data in a reasonable amount of time[2].

We next describe the language-independent method to handle rare words, which is important for training better PCFG-LA grammars especially when the training data is limited in size, and our unknown Chinese word handling method, highlighting the importance of utilizing language-specific features to enhance parsing performance. As we will see later, both of these methods significantly improve parsing performance.

2.1 Rare Word Handling

Whereas rule expansions are frequently observed in the treebank, word-tag co-occurrences are sparser and more likely to suffer from over-fitting. Although the lexicon smoothing method in the Berkeley parser is able to make the word emission probabilities of different latent states of a POS tag more alike, the EM training algorithm still strongly discriminates among word identities. Suppose word tag pairs $\langle w_1, t\rangle$ and $\langle w_2, t\rangle$ both appear the same number of times in the training data. In a PCFG grammar without latent annotations, the probabilities of emitting these two words given tag t would be the same, i.e., $p(w_1|t) = p(w_2|t)$. After introducing latent annotation x to tag t, the emission probabilities of these two words given a latent state t_x may no longer be the same because $p(w_1|t_x)$ and $p(w_2|t_x)$ are two independent parameters that the EM algorithm optimizes on. It is beneficial to learn subcategories of POS tags to model different types of words, especially for frequent words; however, it is not desirable to strongly discriminate among rare words because it could distract the model from learning about common phenomena.

To handle this problem, the probability of a latent state t_x generating a rare word w is forced to be proportional to the emission probability of word w given the surface tag t. This is achieved by mapping all words with frequency less than threshold[3] λ to the *unk* symbol, and for each latent state t_x of a POS tag t, accumulating the word tag statistics of these rare words to $c_r(t_x, \textit{unk}) = \sum_{w:c(w)<\lambda} c(t_x, w)$, and then redistributing them among the rare words to estimate their emission probabilities:

$$c(t_x, w) = c_r(t_x, unk) \cdot \frac{c(t, w)}{c_r(t, unk)}$$
$$p(w|t_x) = c(t_x, w) / \sum\nolimits_w c(t_x, w)$$

2.2 Chinese Unknown Word Handling

The Berkeley parser utilizes statistics associated with rare words (e.g., suffix, capitalization) to estimate the emission probabilities of unknown words at decoding time. This is adequate for for English, however, only a limited number of classes of unknown words, such as digits and dates, are handled for Chinese. In this paper, we develop a character-based unknown word model inspired by (Huang et al., 2007) that reflects the fact that characters in any position (prefix, infix, or suffix) can be predictive of the part-of-speech (POS) type for Chinese words. In our model, the word emission probability, $p(w|t_x)$, of an unknown word w given the latent state t_x of POS tag t is estimated by the geometric average of the emission probability of the characters c_k in the word:

$$P(w|t_x) = \sqrt[n]{\prod\nolimits_{c_k \in w, P(c_k|t) \neq 0} P(c_k|t)}$$

where $n = |\{c_k \in w | P(c_k|t) \neq 0\}|$. Characters not seen in the training data are ignored in the computation of the geometric average. We back off to use the rare word statistics regardless of word identity when the above equation cannot be used to compute the emission probability.

3 Parser Self-Training

Our hypothesis is that combining automatically labeled parses with treebank trees will help the EM training of the PCFG-LA parser to make more informed decisions about latent annotations and thus generate more effective grammars. In this section, we discuss how self-training is applied to train a PCFG-LA parser.

There are several ways to automatically label the data. A fairly standard method is to parse the unlabeled sentences with a parser trained on labeled training data, and then combine the resulting parses with the treebank training data to retrain the parser. This is the approach we chose for self-training. An alternative approach is to run EM directly on the labeled treebank trees and the unlabeled sentences, without explicit parse trees for the unlabeled sentences. However, because the

[2]The parallel version is able to train our largest grammar on a 8-core machine within a week, while the non-parallel version is not able to finish even after 3 weeks.

[3]The value of λ is tuned on the development set.

brackets would need to be determined for the unlabeled sentences together with the latent annotations, this would increase the running time from linear in the number of expansion rules to cubic in the length of the sentence.

Another important decision is how to weight the gold standard and automatically labeled data when training a new parser model. Errors in the automatically labeled data could limit the accuracy of the self-trained model, especially when there is a much greater quantity of automatically labeled data than the gold standard training data. To balance the gold standard and automatically labeled data, one could duplicate the treebank data to match the size of the automatically labeled data; however, the training of the PCFG-LA parser would result in redundant applications of EM computations over the same data, increasing the cost of training. Instead we weight the posterior probabilities computed for the gold and automatically labeled data, so that they contribute equally to the resulting grammar. Our preliminary experiments show that balanced weighting is effective, especially for Chinese (about 0.4% absolute improvement) where the automatic parse trees have a relatively lower accuracy.

The training procedure of the PCFG-LA parser gradually introduces more latent annotations during each split-merge stage, and the self-labeled data can be introduced at any of these stages. Introduction of the self-labeled data in later stages, after some important annotations are learned from the treebank, could result in more effective learning. We have found that a middle stage introduction (after 3 split-merge iterations) of the automatically labeled data has an effect similar to balancing the weights of the gold and automatically labeled trees, possibly due to the fact that both methods place greater trust in the former than the latter. In this study, we introduce the automatically labeled data at the outset and weight it equally with the gold treebank training data in order to focus our experiments to support a deeper analysis.

4 Experimental Setup

For the English experiments, sections from the WSJ Penn Treebank are used as labeled training data: section 2-19 for training, section 22 for development, and section 23 as the test set. We also used 210k[4] sentences of unlabeled news articles in the BLLIP corpus for English self-training.

For the Chinese experiments, the Penn Chinese Treebank 6.0 (CTB6) (Xue et al., 2005) is used as labeled data. CTB6 includes both news articles and transcripts of broadcast news. We partitioned the news articles into train/development/test sets following Huang et al. (2007). The broadcast news section is added to the training data because it shares many of the characteristics of newswire text (e.g., fully punctuated, contains nonverbal expressions such as numbers and symbols). In addition, 210k sentences of unlabeled Chinese news articles are used for self-training. Since the Chinese parsers in our experiments require word-segmented sentences as input, the unlabeled sentences need to be word-segmented first. As shown in (Harper and Huang, 2009), the accuracy of automatic word segmentation has a great impact on Chinese parsing performance. We chose to use the Stanford segmenter (Chang et al., 2008) in our experiments because it is consistent with the treebank segmentation and provides the best performance among the segmenters that were tested. To minimize the discrepancy between the self-training data and the treebank data, we normalize both CTB6 and the self-training data using UW Decatur (Zhang and Kahn, 2008) text normalization.

Table 1 summarizes the data set sizes used in our experiments. We used slightly modified versions of the treebanks; empty nodes and nonterminal-yield unary rules[5], e.g., NP→VP, are deleted using tsurgeon (Levy and Andrew, 2006).

	Train	**Dev**	**Test**	**Unlabeled**
English	39.8k (950.0k)	1.7k (40.1k)	2.4k (56.7k)	210k (5,082.1k)
Chinese	24.4k (678.8k)	1.9k (51.2k)	2.0k (52.9k)	210k (6,254.9k)

Table 1: The number of sentences (and words in parentheses) in our experiments.

We trained parsers on 20%, 40%, 60%, 80%, and 100% of the treebank training data to evaluate

[4]This amount was constrained based on both CPU and memory. We plan to investigate cloud computing to exploit more unlabeled data.

[5]As nonterminal-yield unary rules are less likely to be posited by a statistical parser, it is common for parsers trained on the standard Chinese treebank to have substantially higher precision than recall. This gap between bracket recall and precision is alleviated without loss of parse accuracy by deleting the nonterminal-yield unary rules. This modification similarly benefits both parsers we study here.

the effect of the amount of labeled training data on parsing performance. We also compare how self-training impacts the models trained with different amounts of gold-standard training data. This allows us to simulate scenarios where the language has limited human-labeled resources.

We compare models trained only on the gold labeled training data with those that utilize additional unlabeled data. Self-training (PCFG-LA or Charniak) proceeds in two steps. In the first step, the parser is first trained on the allocated labeled training data (e.g., 40%) and is then used to parse the unlabeled data. In the second step, a new parser is trained on the weighted combination[6] of the allocated labeled training data and the additional automatically labeled data. The development set is used in each step to select the grammar order with the best accuracy for the PCFG-LA parser and to tune the smoothing parameters for Charniak's parser.

5 Results

In this section, we first present the effect of unknown and rare word handling for the PCFG-LA parser, and then compare and discuss the performance of the PCFG-LA parser and Charniak's parser across languages with different amounts of labeled training, either with or without self-training.

5.1 Rare and Unknown Word Handling

Table 2 reports the effect of unknown and rare word handing for the PCFG-LA parser trained on 100%[7] of the labeled training data. The rare word handling improves the English parser by 0.68% and the Chinese parser by 0.56% over the Berkeley parser. The Chinese unknown word handling method alone improves the Chinese parser by 0.47%. The rare and unknown handling methods together improve the Chinese parser by 0.92%. All the improvements are statistically significant[8].

We found that the rare word handling method becomes more effective as the number of latent annotations increases, especially when there is not a sufficient amount of labeled training data. Sharing statistics of the rare words during training results in more robust grammars with better parsing performance. The unknown word handling method also gives greater improvements on grammars trained on smaller amounts of training data, suggesting that it is quite helpful for modeling unseen words at decoding time. However, it tends to be less effective when the number of latent annotations increases, probably because the probability estimation of unseen words based on surface tags is less reliable for finer-gained latent annotations.

	English	Chinese
PCFG-LA	89.95	83.23
+R	90.63	83.79
+U	N/A	83.70
+R+U	N/A	84.15

Table 2: Effects of rare word handling (+R) and Chinese unknown handling (+U) on the test set.

5.2 Labeled Data Only

When comparing the two parsers on both languages in Figure 2 with treebank training, it is clear that they perform much better on English than Chinese. While this is probably due in part to the years of research on English, Chinese still appears to be more challenging than English. The comparison between the two parsing approaches provides two interesting conclusions.

First, the PCFG-LA parser always performs significantly better than Charniak's parser on Chinese, although both model English well. Admittedly Charniak's parser has not been optimized[9] on Chinese, but neither has the PCFG-LA parser[10]. The lexicalized model in Charniak's parser was first optimized for English and required sophisticated smoothing to deal with sparseness; however, the lexicalized model developed for Chinese works less well. In contrast, the PCFG-LA parser learns the latent annotations from the data, without any specification of what precisely should be modeled and how it should be modeled. This flexibility may help it better model new languages.

Second, while both parsers benefit from increased amounts of gold standard training data, the PCFG-LA parser gains more. The PCFG-LA parser is initially poorer than Charniak's parser

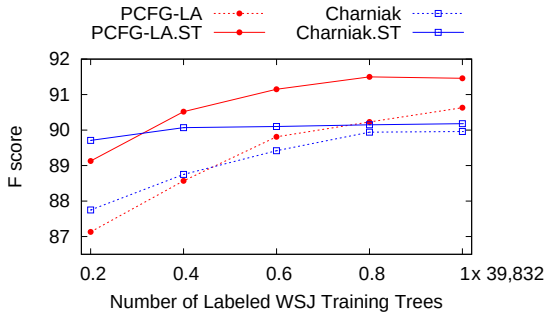
[6] We balance the size of manually and automatically labeled data by posterior weighting for the PCFG-LA parsers and by duplication for Charniak's parser.

[7] Greater improvements are obtained using smaller amounts of labeled training data.

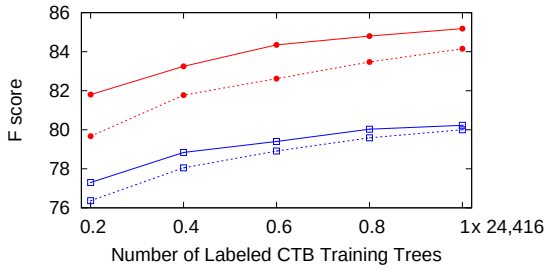
[8] We use Bikel's randomized parsing evaluation comparator to determine the significance ($p < 0.05$) of difference between two parsers' output.

[9] The Chinese port includes modification of the head table, implementation of a Chinese punctuation model, etc.

[10] The PCFG-LA parser without the unknown word handling method still outperforms Charniak's parser on Chinese.



(a) English



(b) Chinese

Figure 2: The performance of the PCFG-LA parser and Charniak's parser evaluated on the test set, trained with different amounts of labeled training data, with and without self-training (ST).

when trained on 20% WSJ training data, probably because the training data is too small for it to learn fine-grained annotations without over-fitting. As more labeled training data becomes available, the performance of the PCFG-LA parser improves quickly and finally outperforms Charniak's parser significantly. Moreover, performance of the PCFG-LA parser continues to grow when more labeled training data is available, while the performance of Charniak's parser levels out at around 80% of the labeled data. The PCFG-LA parser improves by 3.5% when moving from 20% to 100% training data, compared to a 2.21% gain for Charniak's parser. Similarly for Chinese, the PCFG-LA parser also gains more (4.48% vs 3.63%).

5.3 Labeled + Self-Labeled

The PCFG-LA parser is also able to benefit more from self-training than Charniak's parser. On the WSJ data set, Charniak's parser benefits from self-training initially when there is little labeled training data, but the improvement levels out quickly as more labeled training trees become available. In contrast, the PCFG-LA parser benefits consistently from self-training[11], even when using 100% of the labeled training set. Similar trends are also found for Chinese.

It should be noted that the PCFG-LA parser trained on a fraction of the treebank training data plus a large amount of self-labeled training data, which comes with little or no cost, performs comparably or even better than grammars trained with additional labeled training data. For example, the self-trained PCFG-LA parser with 60% labeled data is able to outperform the grammar trained with 100% labeled training data alone for both English and Chinese. With self-training, even 40% labeled WSJ training data is sufficient to train a PCFG-LA parser that is comparable to the model trained on the entire WSJ training data alone. This is of significant importance, especially for languages with limited human-labeled resources.

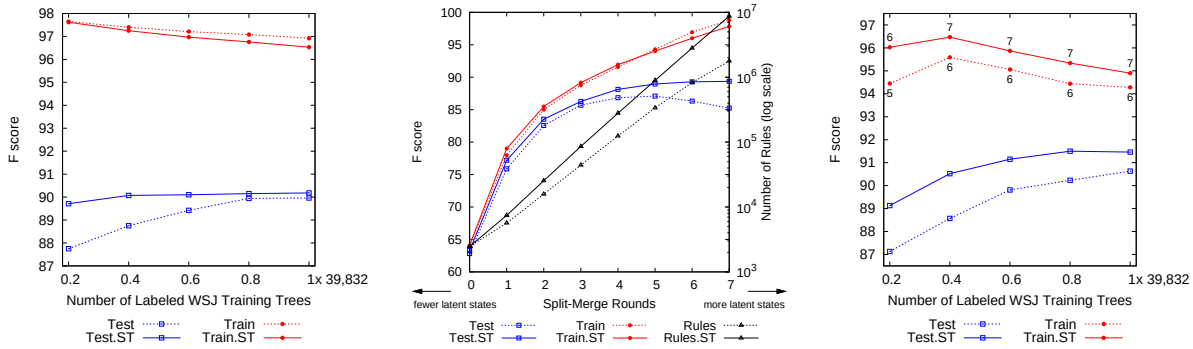
One might conjecture that the PCFG-LA parser benefits more from self-training than Charniak's parser because its self-labeled data has higher accuracy. However, this is not true. As shown in Figure 2 (a), the PCFG-LA parser trained with 40% of the WSJ training set alone has a much lower performance (88.57% vs 89.96%) than Charniak's parser trained on the full WSJ training set. With the same amount of self-training data (labeled by each parser), the resulting PCFG-LA parser obtains a much higher F score than the self-trained Charniak's parser (90.52% vs 90.18%). Similar patterns can also be found for Chinese.

	English	Chinese
PCFG-LA	90.63	84.15
+ Self-training	91.46	85.18

Table 3: Final results on the test set.

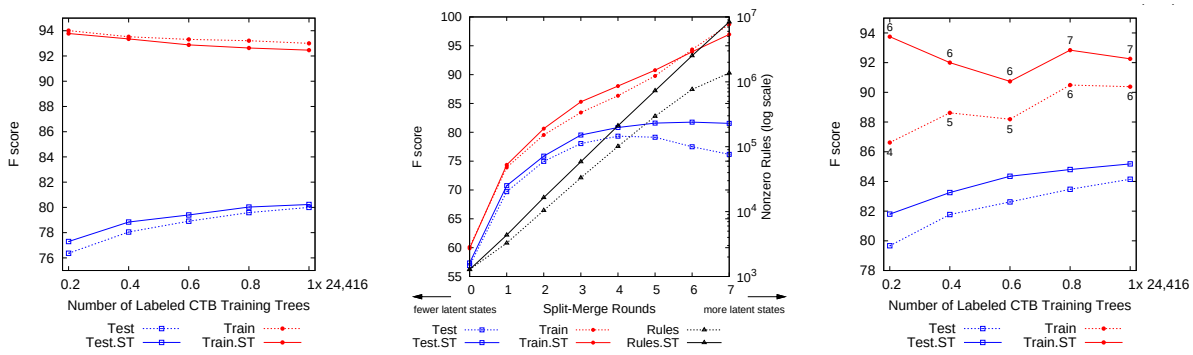
Table 3 reports the final results on the test set when trained on the entire WSJ or CTB6 training set. For English, self-training contributes 0.83% absolute improvement to the PCFG-LA parser, which is comparable to the improvement obtained from using semi-supervised training with the two-stage parser in (McClosky et al., 2006). Note that their improvement is achieved with the addition of 2,000k unlabeled sentences using the combination of a generative parser and a discriminative reranker, compared to using only 210k unlabeled sentences with a single generative parser in our approach. For Chinese, self-training results in a

[11] One may notice that the self-trained PCFG-LA parser with 100% labeled WSJ data has a slightly lower test accuracy than the self-trained PCFG-LA parser with 80% labeled WSJ data. This is due to the variance in parser performance when initialized with different seeds and the fact that the development set is used to pick the best model for evaluation.



(a) Charniak (b) PCFG-LA (20% WSJ) (c) PCFG-LA

Figure 3: (a) The training/test accuracy of Charniak's parser trained on varying amounts of labeled WSJ training data, with and without self-training (ST). (b) The training/test accuracy and the number of nonzero rules of the PCFG-LA grammars trained on 20% of the labeled WSJ training data, w/ and w/o ST. (c) The training/test accuracy of the PCFG-LA parser trained on varying amount of labeled WSJ training data, w/ and w/o ST; the numbers along the training curves indicate the order of the grammars.



(a) Charniak (b) PCFG-LA (20% CTB) (c) PCFG-LA

Figure 4: (a) The training/test accuracy of Charniak's parser trained on varying amounts of labeled CTB training data, with and without self-training (ST). (b) The training/test accuracy and the number of nonzero rules of the PCFG-LA grammars trained on 20% of the labeled CTB training data, w/ and w/o ST. (c) The training/test accuracy of the PCFG-LA parser trained on varying amount of labeled CTB training data, w/ and w/o ST; the numbers along the training curves indicate the order of the grammars.

state-of-the-art parsing model with 85.18% accuracy (1.03% absolute improvement) on a representative test set. Both improvements are statistically significant.

6 Analysis

In this section, we perform a series of analyses, focusing on English (refer to Figure 3), to investigate why the PCFG-LA parser benefits more from additional data, most particularly automatically labeled data, when compared to Charniak's parser. Similar analyses have been done for Chinese with similar results (refer to Figure 4).

Charniak's parser is a lexicalized PCFG parser that models lexicalized dependencies explicitly observable in the training data and relies on smoothing to avoid over-fitting. Although it is able to benefit from more training data because of broader lexicon and rule coverage and more robust estimation of parameters, its ability to benefit from the additional data is limited in the sense that it is not able to generate additional predictive features that are supported by this data. As shown in figure 3(a), the parsing accuracy of Charniak's parser on the test set improves as the amount of labeled training data increases; however, the training accuracy[12] degrades as more data is added. Note that the training accuracy[13] of Charniak's parser also

[12]The parser is tested on the treebank labeled set that the parser is trained on.

[13]The self-training data is combined with the labeled treebank trees in a weighted manner; otherwise, the training accuracy would be even lower.

decreases after the addition of self-training data. This is expected for models like Charniak's parser with fixed model parameters; it is harder to model more data with greater diversity. The addition of self-labeled data helps on the test set initially but it provides little gain when the labeled training data becomes relatively large.

In contrast, the PCFG-LA grammar is able to model the training data with different granularities. Fewer latent annotations are employed when the training set is small. As the size of the training data increases, it is able to allocate more latent annotations to better model the data. As shown in Figure 3 (b), for a fixed amount (20%) of labeled training data, the accuracy of the model on training data continues to improve as the number of latent annotation increases. Although it is important to limit the number of latent annotations to avoid over-fitting, the ability to model training data accurately given sufficient latent annotations is desirable when more training data is available. When trained on the labeled data (20%) alone, the 5-th order grammar achieves its optimal generalization performance (based on the development set) and begins to degrade afterwords. With the addition of self-training data, the 5-th order grammar achieves an even greater accuracy on the test set and its performance continues to increase[14] when moving to the 6-th or even 7-th order grammar.

Figure 3 (c) plots the training and test curves of the English PCFG-LA parser with varying amounts of labeled training data, with and without self-training. This figure differs substantially from Figure 3 (a). First, as mentioned earlier, the PCFG-LA parser benefits much more from self-training than Charniak's parser with moderate to large amounts of labeled training data. Second, in contrast to Charniak's parser for which training accuracy degrades consistently as the amount of labeled training data increases, the training accuracy of the PCFG-LA parser sometimes improves when trained on more labeled training data (e.g., the best model (at order 6) trained on 40%[15] labeled training data alone has a higher training accuracy than the best model (at order 5) trained on 20% labeled training data). Third, the addition of self-labeled data supports more accurate PCFG-LA grammars with higher orders than those trained without self-training, as evidenced by scores on both the training and test data. This suggests that the self-trained grammars are able to utilize more latent annotations to learn deeper dependencies.

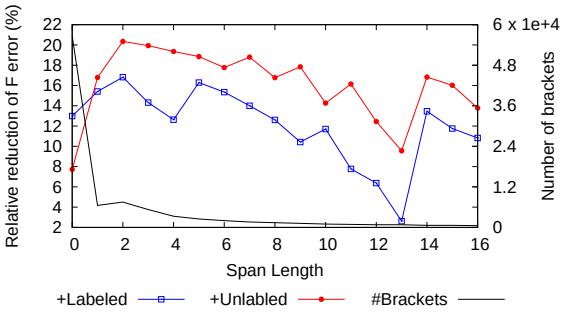


Figure 5: The relative reduction of bracketing errors for different span lengths, evaluated on the test set. The baseline model is the PCFG-LA parser trained on 20% of the WSJ training data. The +Unlabeled curve corresponds to the parser trained with the additional automatically labeled data and the +Labeled curve corresponds to the parser trained with additional 20% labeled training data. The counts of the brackets are computed on the gold reference. Span length '0' is designated for the effect on preterminal POS tags to differentiate it from the non-terminal brackets spanning only one word.

Figure 5 compares the effect of additional treebank labeled and automatically labeled data on the relative reduction of bracketing errors for different span lengths. It is clear from the figure that the improvement in parsing accuracy from self-training is the result of better bracketing across all span lengths[16]. However, even though the automatically labeled training data provides more improvement than the additional treebank labeled data in terms of parsing accuracy, this data is less effective at improving tagging accuracy than the additional treebank labeled training data.

So, how could self-training improve rule estimation when training the PCFG-LA parser with more latent annotations? One possibility is that the automatically labeled data smooths the parameter

[14] Although the 20% self-trained grammar has a higher test accuracy at the 7-th round than the 6-th round, the development accuracy was better at the 6-th round, and thus we report the test accuracy of the 6-th round grammar in Figure 3 (c).

[15] For models trained with greater amounts of labeled training data, although their training accuracy becomes lower (due to greater diversity) for the grammars (all at order 6) selected by the development set, their 7-th order grammars (not reported in the figure) actually have both higher training and test accuracies than the 6-th order grammar trained on less training data.

[16] There is a slight degradation in bracketing accuracy for some spans longer than 16 words, but the effect is negligible due to their low counts.

estimates in the EM algorithm, enabling effective training of models with more parameters to learn deeper dependencies. Let $p(a \rightarrow b|e,t)$ be the posterior probability of expanding subcategories a to b given the event e, which is a rule expansion on a treebank parse tree t. T_l and T_u are the sets of gold and automatically labeled parse trees, respectively. The update of the rule expansion probability $p(a \rightarrow b)$ in self-training (with weighting parameter α) can be expressed as:

$$\frac{\sum_{t \in T_l} \sum_{e \in t} p(a \rightarrow b|e,t) + \alpha \sum_{t \in T_u} \sum_{e \in t} p(a \rightarrow b|e,t)}{\sum_{b} (\sum_{t \in T_l} \sum_{e \in t} p(a \rightarrow b|e,t) + \alpha \sum_{t \in T_u} \sum_{e \in t} p(a \rightarrow b|e,t))}$$

Since the unlabeled data is parsed by a lower order grammar (with fewer latent annotations), the expected counts from the automatically labeled data can be thought of as counts from a lower-order grammar[17] that smooth the higher-order (with more latent annotations) grammar.

We observe that many of the rule parameters of the grammar trained on WSJ training data alone have zero probabilities (rules with extremely low probabilities are also filtered to zero), as was also pointed out in (Petrov et al., 2006). On the one hand, this is what we want because the grammar should learn to avoid impossible rule expansions. On the other hand, this might also be a sign of over-fitting of the labeled training data. As shown in Figure 3 (b), the grammar obtained with the addition of automatically labeled data contains many more non-zero rules, and its performance continues to improve with more latent annotations. Similar patterns also appear when using self-training for other amounts of labeled training data. As is partially reflected by the zero probability rules, the addition of the automatically labeled data enables the exploration of a broader parameter space with less danger of over-fitting the data. Also note that the benefit of the automatically labeled data is less clear in the early training stages (i.e., when there are fewer latent annotations), as can be seen in Figure 3 (b). This is probably because there is a small number of free parameters and the treebank data is sufficiently large for robust parameter estimation.

[17] We also trained models using only the automatically labeled data without combining it with human-labeled training data, but they were no more accurate than those trained on the human-labeled training data alone without self-training.

7 Conclusion

In this paper, we showed that PCFG-LA parsers can be more effectively applied to languages where parsing is less well developed and that they are able to benefit more from self-training than lexicalized generative parsers. We show for the first time that self-training is able to significantly improve the performance of a PCFG-LA parser, a single generative parser, on both small and large amounts of labeled training data.

We conjecture based on our analysis that the EM training algorithm is able to exploit the information available in both gold and automatically labeled data with more complex grammars while being less affected by over-fitting. Better results would be expected by combining the PCFG-LA parser with discriminative reranking approaches (Charniak and Johnson, 2005; Huang, 2008) for self training. Self-training should also benefit other discriminatively trained parsers with latent annotations (Petrov and Klein, 2008), although training would be much slower compared to using generative models, as in our case.

In future work, we plan to scale up the training process with more unlabeled training data (e.g., gigaword) and investigate automatic selection of materials that are most suitable for self-training. We also plan to investigate domain adaptation and apply the model to other languages with modest treebank resources. Finally, it is also important to explore other ways to exploit the use of unlabeled data.

Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023 and NSF IIS-0703859. Any opinions, findings and/or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the institutions where the work was completed.

References

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank. In *Proceedings of the Second Chinese Language Processing Workshop*.

Pi-Chuan Chang, Michel Gally, and Christopher Manning. 2008. Optimizing chinese word segmentation

for machine translation performance. In *ACL 2008 Third Workshop on Statistical Machine Translation*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *ICAI*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *ACL*.

Michael John Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Mary Harper and Zhongqiang Huang. 2009. Chinese statistical parsing. To appear in *The Gale Book*.

Zhongqiang Huang, Mary Harper, and Wen Wang. 2007. Mandarin part-of-speech tagging and discriminative reranking. In *EMNLP*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*.

Terry Koo, Xavier Carrera, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL*.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: Tools for querying and manipulating tree data structures. In *LREC*.

Roger Levy and Christopher Manning. 2003. Is it harder to parse chinese, or the chinese treebank. In *ACL*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *HLT-NAACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *COLING*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*.

Slav Petrov and Dan Klein. 2008. Sparse multi-scale grammars for discriminative latent variable parsing. In *EMNLP*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *ACL*.

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

Qin Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *ACL*.

Nianwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*.

Bin Zhang and Jeremy G. Kahn. 2008. Evaluation of decatur text normalizer for language model training. Technical report, University of Washington.