Self-training with Products of Latent Variable Grammars

Zhongqiang Huang[†]
[†]UMIACS
University of Maryland
College Park, MD
zqhuang@umd.edu

Mary Harper[†‡]
[‡]HLT Center of Excellence
Johns Hopkins University
Baltimore, MD
mharper@umd.edu

Slav Petrov[*]
[*]Google Research
76 Ninth Avenue
New York, NY
slav@google.com

Abstract

We study self-training with products of latent variable grammars in this paper. We show that increasing the quality of the automatically parsed data used for self-training gives higher accuracy self-trained grammars. Our generative self-trained grammars reach F scores of 91.6 on the WSJ test set and surpass even discriminative reranking systems without self-training. Additionally, we show that multiple self-trained grammars can be combined in a product model to achieve even higher accuracy. The product model is most effective when the individual underlying grammars are most diverse. Combining multiple grammars that were self-trained on disjoint sets of unlabeled data results in a final test accuracy of 92.5% on the WSJ test set and 89.6% on our Broadcast News test set.

1 Introduction

The latent variable approach of Petrov et al. (2006) is capable of learning high accuracy context-free grammars directly from a raw treebank. It starts from a coarse treebank grammar (Charniak, 1997), and uses latent variables to refine the context-free assumptions encoded in the grammar. A hierarchical split-and-merge algorithm introduces grammar complexity gradually, iteratively splitting (and potentially merging back) each observed treebank category into a number of increasingly refined latent subcategories. The Expectation Maximization (EM) algorithm is used to train the model, guaranteeing that each EM iteration will increase the training likelihood. However, because the latent variable grammars are not explicitly regularized, EM keeps fitting the training data and eventually begins overfitting (Liang et al., 2007). Moreover, EM is a local method, making no promises regarding the final point of convergence when initialized from different random seeds. Recently, Petrov (2010) showed that substantial differences between the learned grammars remain, even if the hierarchical splitting reduces the variance across independent runs of EM.

In order to counteract the overfitting behavior, Petrov et al. (2006) introduced a linear smoothing procedure that allows training grammars for 6 split-merge (SM) rounds without overfitting. The increased expressiveness of the model, combined with the more robust parameter estimates provided by the smoothing, results in a nice increase in parsing accuracy on a held-out set. However, as reported by Petrov (2009) and Huang and Harper (2009), an additional 7th SM round actually hurts performance.

Huang and Harper (2009) addressed the issue of data sparsity and overfitting from a different angle. They showed that self-training latent variable grammars on their own output can mitigate data sparsity issues and improve parsing accuracy. Because the capacity of the model can grow with the size of the training data, latent variable grammars are able to benefit from the additional training data, even though it is not perfectly labeled. Consequently, they also found that a 7th round of SM training was beneficial in the presence of large amounts of training data. However, variation still remains in their self-trained grammars and they had to use a held-out set for model selection.

The observation of variation is not surprising; EM's tendency to get stuck in local maxima has been studied extensively in the literature, resulting in various proposals for model selection methods (e.g., see

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 12–22,
MIT, Massachusetts, USA, 9-11 October 2010.

Burnham and Anderson (2002)). What is perhaps more surprising is that the different latent variable grammars seem to capture complementary aspects of the data. Petrov (2010) showed that a simple randomization scheme produces widely varying grammars. Quite serendipitously, these grammars can be combined into an unweighted product model that substantially outperforms the individual grammars.

In this paper, we combine the ideas of self-training and product models and show that both techniques provide complementary effects. We hypothesize that the main factors contributing to the final accuracy of the product model of self-trained grammars are (i) the accuracy of the grammar used to parse the unlabeled data for retraining (single grammar versus product of grammars) and (ii) the diversity of the grammars that are being combined (self-trained grammars trained using the same automatically labeled subset or different subsets). We conduct a series of analyses to develop an understanding of these factors, and conclude that both dimensions are important for obtaining significant improvements over the standard product models.

2 Experimental Setup

2.1 Data

We conducted experiments in two genres: newswire text and broadcast news transcripts. For the newswire studies, we used the standard setup (sections 02-21 for training, 22 for development, and 23 for final test) of the WSJ Penn Treebank (Marcus et al., 1999) for supervised training. The BLLIP corpus (Charniak et al., 2000) was used as a source of unlabeled data for self-training the WSJ grammars. We ignored the parse trees contained in the BLLIP corpus and retained only the sentences, which are already segmented and tokenized for parsing (e.g., contractions are split into two tokens and punctuation is separated from the words). We partitioned the 1,769,055 BLLIP sentences into 10 equally sized subsets[1].

For broadcast news (BN), we utilized the Broadcast News treebank from Ontonotes (Weischedel et al., 2008) together with the WSJ Penn Treebank for supervised training, because their combination results in better parser models compared to using the limited-sized BN corpus alone (86.7 F vs. 85.2 F). The files in the Broadcast News treebank represent news stories collected during different time periods with a diversity of topics. In order to obtain a representative split of train-test-development sets, we divided them into blocks of 10 files sorted by alphabetical filename order. We used the first file in each block for development, the second for test, and the remaining files for training. This training set was then combined with the entire WSJ treebank. We also used 10 equally sized subsets from the Hub4 CSR 1996 utterances (Garofolo et al., 1996) for self-training. The Hub 4 transcripts are markedly noisier than the BLLIP corpus is, in part because it is harder to sentence segment, but also because it was produced by human transcription of spoken language.

The treebanks were pre-processed differently for the two genres. For newswire, we used a slightly modified version of the WSJ treebank: empty nodes and function labels were deleted and auxiliary verbs were replaced with AUXB, AUXG, AUXZ, AUXD, or AUXN to represent infinitive, progressive, present, past, or past participle auxiliaries[2]. The targeted use of the broadcast models is for parsing broadcast news transcripts for language models in speech recognition systems. Therefore, in addition to applying the transformations used for newswire, we also replaced symbolic expressions with verbal forms (e.g., $5 was replaced with five dollars) and removed punctuation and case. The Hub4 data was segmented into utterances, punctuation was removed, words were down-cased, and contractions were tokenized for parsing. Table 1 summarizes the data set sizes used in our experiments, together with average sentence length and standard deviation.

2.2 Scoring

Parses from all models are compared with respective gold standard parses using SParseval bracket scoring (Roark et al., 2006). This scoring tool pro-

[1] We corrected some of the most egregious sentence segmentation problems in this corpus, and so the number of sentences is different than if one simply pulled the fringe of the trees. It was not uncommon for a sentence split to occur on abbreviations, such as Adm.

[2] Parsing accuracy is marginally affected. The average over 10 SM6 grammars with the transformation is 90.5 compared to 90.4 F without it, a 0.1% average improvement.

Genre	Statistics	Train	Dev	Test	Unlabeled
Newswire	# sentences	39.8k	1.7k	2.4k	1,769.1k
	# words	950.0k	40.1k	56.7k	43,057.0k
	length Avg./Std.	28.9/11.2	25.1/11.8	25.1/12.0	24.3/10.9
Broadcast News	# sentences	59.0k	1.0k	1.1k	4,386.5k
	# words	1,281.1k	17.1k	19.4k	77,687.9k
	length Avg./Std.	17.3/11.3	17.4/11.3	17.7/11.4	17.7/12.8

Table 1: The number of words and sentences, together with average (Avg.) sentence length and its standard deviation (Std.), for the data sets used in our experiments.

duces scores that are identical to those produced by EVALB for WSJ. For Broadcast News, SParseval applies Charniak and Johnson's (Charniak and Johnson, 2001) scoring method for EDITED nodes[3]. Using this method, BN scores were slightly (.05-.1) lower than if EDITED constituents were treated like any other, as in EVALB.

2.3 Latent Variable Grammars

We use the latent variable grammar (Matsuzaki et al., 2005; Petrov et al., 2006) implementation of Huang and Harper (2009) in this work. Latent variable grammars augment the observed parse trees in the treebank with a latent variable at each tree node. This effectively splits each observed category into a set of latent subcategories. An EM-algorithm is used to fit the model by maximizing the joint likelihood of parse trees and sentences. To allocate the grammar complexity only where needed, a simple split-and-merge procedure is applied. In every split-merge (SM) round, each latent variable is first split in two and the model is re-estimated. A likelihood criterion is used to merge back the least useful splits (50% merge rate for these experiments). This iterative refinement proceeds for 7 rounds, at which point parsing performance on a held-out set levels off and training becomes prohibitively slow.

Since EM is a local method, different initializations will result in different grammars. In fact, Petrov (2010) recently showed that this EM-algorithm is very unstable and converges to widely varying local maxima. These local maxima correspond to different high quality latent variable grammars that have captured different types of patterns in the data. Because the individual models' mistakes are independent to some extent, multiple grammars can be effectively combined into an unweighted product model of much higher accuracy. We build upon this line of work and investigate methods to exploit products of latent variable grammars in the context of self-training.

[3]Non-terminal subconstituents of EDITED nodes are removed so that the terminal constituents become immediate children of a single EDITED node, adjacent EDITED nodes are merged, and they are ignored for span calculations of the other constituents.

3 Self-training Methodology

Different types of parser self-training have been proposed in the literature over the years. All of them involve parsing a set of unlabeled sentences with a baseline parser and then estimating a new parser by combining this automatically parsed data with the original training data. McClosky et al. (2006) presented a very effective method for self-training a two-stage parsing system consisting of a first-stage generative lexicalized parser and a second-stage discriminative reranker. In their approach, a large amount of unlabeled text is parsed by the two-stage system and the parameters of the first-stage lexicalized parser are then re-estimated taking the counts from the automatically parsed data into consideration.

More recently Huang and Harper (2009) presented a self-training procedure based on an EM-algorithm. They showed that the EM-algorithm that is typically used to fit a latent variable grammar (Matsuzaki et al., 2005; Petrov et al., 2006) to a treebank can also be used for self-training on automatically parsed sentences. In this paper, we investigate self-training with products of latent variable grammars. We consider three training scenarios:

ST-Reg Training Use the best single grammar to

Regular	Best	Average	Product
SM6	90.8	90.5	92.0
SM7	90.4	90.1	92.2

Table 2: Performance of the regular grammars and their products on the WSJ development set.

ST-Reg	Best	Average	Product
SM6	91.5	91.2	92.0
SM7	91.6	91.5	92.4

Table 3: Performance of the ST-Reg grammars and their products on the WSJ development set.

parse a single subset of the unlabeled data and train 10 self-trained grammars using this single set.

ST-Prod Training Use the product model to parse a single subset of the unlabeled data and train 10 self-trained grammars using this single set.

ST-Prod-Mult Training Use the product model to parse all 10 subsets of the unlabeled data and train 10 self-trained grammars, each using a different subset.

The resulting grammars can be either used individually or combined in a product model.

These three conditions provide different insights. The first experiment allows us to investigate the effectiveness of product models for standard self-trained grammars. The second experiment enables us to quantify how important the accuracy of the baseline parser is for self-training. Finally, the third experiment investigates a method for injecting some additional diversity into the individual grammars to determine whether a product model is most successful when there is more variance among the individual models.

Our initial experiments and analysis will focus on the development set of WSJ. We will then follow up with an analysis of broadcast news (BN) to determine whether the findings generalize to a second, less structured type of data. It is important to construct grammars capable of parsing this type of data accurately and consistently in order to support structured language modeling (e.g., (Wang and Harper, 2002; Filimonov and Harper, 2009)).

4 Newswire Experiments

In this section, we compare single grammars and their products that are trained in the standard way with gold WSJ training data, as well as the three self-training scenarios discussed in Section 3. We report the F scores of both SM6 and SM7 grammars on the development set in order to evaluate the effect of model complexity on the performance of the self-trained and product models. Note that we use 6th round grammars to produce the automatic parse trees for the self-training experiments. Parsing with the product of the 7th round grammars is slow and requires a large amount of memory (32GB). Since we had limited access to such machines, it was infeasible for us to parse all of the unlabeled data with the SM7 product grammars.

4.1 Regular Training

We begin by training ten latent variable models initialized with different random seeds using the gold WSJ training set. Results are presented in Table 2. The best F score attained by the individual SM6 grammars on the development set is 90.8, with an average score of 90.5. The product of grammars achieves a significantly improved accuracy at 92.0[4]. Notice that the individual SM7 grammars perform worse on average (90.1 vs. 90.5) due to overfitting, but their product achieves higher accuracy than the product of the SM6 grammars (92.2 vs. 92.0). We will further investigate the causes for this effect in Section 5.

4.2 ST-Reg Training

Given the ten SM6 grammars from the previous subsection, we can investigate the three self-training methods. In the first regime (ST-Reg), we use the best single grammar (90.8 F) to parse a single subset of the BLLIP data. We then train ten grammars from different random seeds, using an equally weighted combination of the WSJ training set with this single set. These self-trained grammars are then combined into a product model. As reported in Table 3,

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

ST-Prod	Best	Average	Product
SM6	91.7	91.4	92.2
SM7	91.9	91.7	92.4

Table 4: Performance of the ST-Prod grammars and their products on the WSJ development set.

ST-Prod-Mult	Best	Average	Product
SM6	91.7	91.4	92.5
SM7	91.8	91.7	92.8

Table 5: Performance of the ST-Prod-Mult grammars and their products on the WSJ development set.

thanks to the use of additional automatically labeled training data, the individual SM6 ST-Reg grammars perform significantly better than the individual SM6 grammars (91.2 vs. 90.5 on average), and the individual SM7 ST-Reg grammars perform even better, achieving a high F score of 91.5 on average.

The product of ST-Reg grammars achieves significantly better performance over the individual grammars, however, the improvement is much smaller than that obtained by the product of regular grammars. In fact, the product of ST-Reg grammars performs quite similarly to the product of regular grammars despite the higher average accuracy of the individual grammars. This may be caused by the fact that self-training on the same data tends to reduce the variation among the self-trained grammars. We will show in Section 5 that the diversity among the individual grammars is as important as average accuracy for the performance attained by the product model.

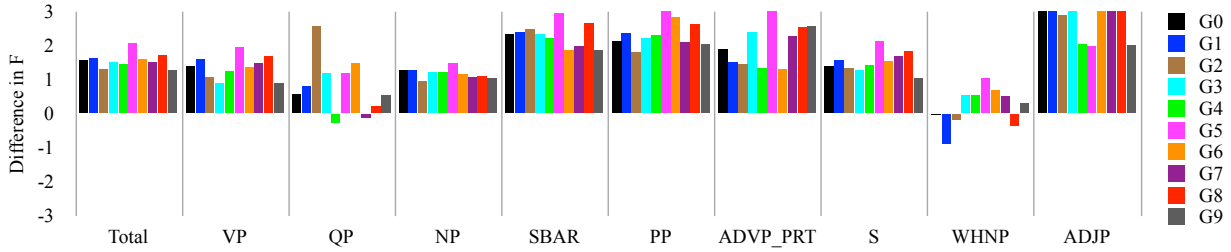
4.3 ST-Prod Training

Since products of latent variable grammars perform significantly better than individual latent variable grammars, it is natural to try using the product model for parsing the unlabeled data. To investigate whether the higher accuracy of the automatically labeled data translates into a higher accuracy of the self-trained grammars, we used the product of 6th round grammars to parse the same subset of the unlabeled data as in the previous experiment. We then trained ten self-trained grammars, which we call ST-Prod grammars. As can be seen in Table 4, using the product of the regular grammars for labeling the self-training data results in improved individual ST-Prod grammars when compared with the ST-Reg grammars, with 0.2 and 0.3 improvements for the best SM6 and SM7 grammars, respectively. Interestingly, the best individual SM7 ST-Prod grammar (91.9 F) performs comparably to the product of the regular grammars (92.0 F) that was used to label the BLLIP subset used for self-training. This is very useful for practical reasons because a single grammar is faster to parse with and requires less memory than the product model.

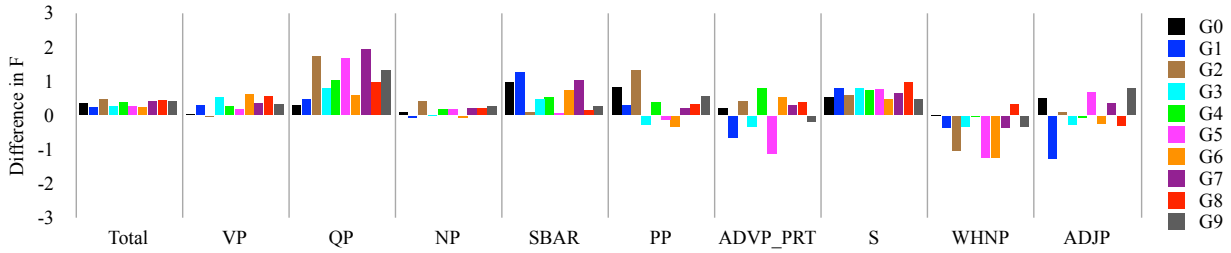
The product of the SM6 ST-Prod grammars also achieves a 0.2 higher F score compared to the product of the SM6 ST-Reg grammars, but the product of the SM7 ST-Prod grammars has the same performance as the product of the SM7 ST-Reg grammars. This could be due to the fact that the ST-Prod grammars are no more diverse than the ST-Reg grammars, as we will show in Section 5.

4.4 ST-Prod-Mult Training

When creating a product model of regular grammars, Petrov (2010) used a different random seed for each model and conjectured that the effectiveness of the product grammars stems from the resulting diversity of the individual grammars. Two ways to systematically introduce bias into individual models are to either modify the feature sets (Baldridge and Osborne, 2008; Smith and Osborne, 2007) or to change the training distributions of the individual models (Breiman, 1996). Petrov (2010) attempted to use the second method to train individual grammars on either disjoint or overlapping subsets of the treebank, but observed a performance drop in individual grammars resulting from training on less data, as well as in the performance of the product model. Rather than reducing the amount of gold training data (or having treebank experts annotate more data to support the diversity), we employ the self-training paradigm to train models using a combination of the same gold training data with different sets of the self-labeled training data. This approach also allows us to utilize a much larger amount of low-cost self-labeled data than can be used to train one model by partitioning the data into ten subsets and then training ten models with a different subset. Hence, in



(a) Difference in F score between the product and the individual SM6 regular grammars.



(b) Difference in F score between the product of SM6 regular grammars and the individual SM7 ST-Prod-Mult grammars.

Figure 1: Difference in F scores between various individual grammars and representative product grammars.

the third self-training experiment, we use the product of the regular grammars to parse all ten subsets of the unlabeled data and train ten grammars, which we call ST-Prod-Mult grammars, each using a different subset.

As shown in Table 5, the individual ST-Prod-Mult grammars perform similarly to the individual ST-Prod grammars. However, the product of the ST-Prod-Mult grammars achieves significantly higher accuracies than the product of the ST-Prod grammars, with 0.3 and 0.4 improvements for SM6 and SM7 grammars, respectively, suggesting that the use of multiple self-training subsets plays an important role in model combination.

5 Analysis

We conducted a series of analyses to develop an understanding of the factors affecting the effectiveness of combining self-training with product models.

5.1 What Has Improved?

Figure 1(a) depicts the difference between the product and the individual SM6 regular grammars on overall F score, as well as individual constituent F scores. As can be observed, there are significant variations among the individual grammars, and the product of the regular grammars improves almost all categories, with a few exceptions (some individual grammars do better on QP and WHNP constituents).

Figure 1(b) shows the difference between the product of the SM6 regular grammars and the individual SM7 ST-Prod-Mult grammars. Self-training dramatically improves the quality of single grammars. In most of the categories, some individual ST-Prod-Mult grammars perform comparably or slightly better than the product of SM6 regular grammars used to automatically label the unlabeled training set.

5.2 Overfitting vs. Smoothing

Figure 2(a) and 2(b) depict the learning curves of the regular and the ST-Prod-Mult grammars. As more latent variables are introduced through the iterative SM training algorithm, the modeling capacity of the grammars increases, leading to improved performance. However, the performance of the regular grammars drops after 6 SM rounds, as also previously observed in (Huang and Harper, 2009; Petrov, 2009), suggesting that the regular SM7 grammars have overfit the relatively small-sized gold training

data. In contrast, the performance of the self-trained grammars continues to improve in the 7th SM round. Huang and Harper (2009) argued that the additional self-labeled training data adds a smoothing effect to the grammars, supporting an increase in model complexity without overfitting.

Although the performance of the individual grammars, both regular and self-trained, varies significantly and the product model consistently helps, there is a non-negligible difference between the improvement achieved by the two product models over their component grammars. The regular product model improves upon its individual grammars more than the ST-Prod-Mult product does in the later SM rounds, as illustrated by the relative error reduction curves in figures 2(a) and (b). In particular, the product of the SM7 regular grammars gains a remarkable 2.1% absolute improvement over the average performance of the individual regular SM7 grammars and 0.2% absolute over the product of the regular SM6 grammars, despite the fact that the individual regular SM7 grammars perform worse than the SM6 grammars. This suggests that the product model is able to effectively exploit less smooth, overfit grammars. We will examine this issue further in the next subsection.

5.3 Diversity

From the perspective of Products of Experts (Hinton, 1999) or Logarithmic Opinion Pools (Smith et al., 2005), each individual expert learns complementary aspects of the training data and the veto power of product models enforces that the joint prediction of their product has to be licensed by all individual experts. One possible explanation of the observation in the previous subsection is that with the addition of more latent variables, the individual grammars become more deeply specialized on certain aspects of the training data. This specialization leads to greater diversity in their prediction preferences, especially in the presence of a small training set. On the other hand, the self-labeled training set size is much larger, and so the specialization process is therefore slowed down.

Petrov (2010) showed that the individually learned grammars are indeed very diverse by looking at the distribution of latent annotations across the treebank categories, as well as the variation in overall and individual category F scores (see Figure 1). However, these measures do not directly relate to the diversity of the prediction preferences of the grammars, as we observed similar patterns in the regular and self-trained models.

Given a sentence s and a set of grammars $\mathcal{G} = \{G_1, \cdots, G_n\}$, recall that the decoding algorithm of the product model (Petrov, 2010) searches for the best tree T such that the following objective function is maximized:

$$\sum_{r \in T} \sum_{G \in \mathcal{G}} \log p(r|s, G)$$

where $\log p(r|s, G)$ is the log posterior probability of rule r given sentence s and grammar G. The power of the product model comes directly from the diversity in $\log p(r|s, G)$ among individual grammars. If there is little diversity, the individual grammars would make similar predictions and there would be little or no benefit from using a product model. We use the average empirical variance of the log posterior probabilities of the rules among the learned grammars over a held-out set S as a proxy of the diversity among the grammars:

$$\frac{\sum_{s \in S} \sum_{G \in \mathcal{G}} \sum_{r \in \mathcal{R}(G,s)} p(r|s, G) \text{VAR}(\log(p(r|s, \mathcal{G})))}{\sum_{s \in S} \sum_{G \in \mathcal{G}} \sum_{r \in \mathcal{R}(G,s)} p(r|s, G)}$$

where $\mathcal{R}(G, s)$ represents the set of rules extracted from the chart when parsing sentence s with grammar G, and $\text{VAR}(\log(p(r|s, \mathcal{G})))$ is the variance of $\log(p(r|s, G))$ among all grammars $G \in \mathcal{G}$.

Note that the average empirical variance is only an approximation of the diversity among grammars. In particular, this measure tends to be biased to produce larger numbers when the posterior probabilities of rules tend to be small, because small differences in probability produce large changes in the log scale. This happens for coarser grammars produced in early SM stages when there is more uncertainty about what rules to apply, with the rules remaining in the parsing chart having low probabilities overall.

As shown in Figure 2(c), the average variances all start at a high value and then drop, probably due to the aforementioned bias. However, as the SM iteration continues, the average variances increase despite the bias. More interestingly, the variance

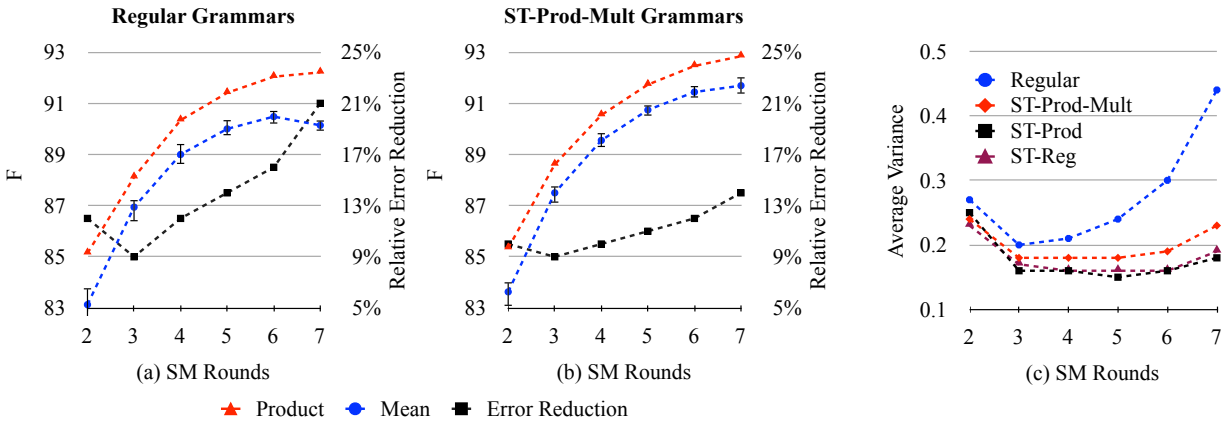


Figure 2: Learning curves of the individual regular (a) and ST-Prod-Mult (b) grammars (average performance, with minimum and maximum values indicated by bars) and their products before and after self-training on the WSJ development set. The relative error reductions of the products are also reported. (c) The measured average empirical variance among the grammars trained on WSJ.

among the regular grammars grows at a much faster speed and is consistently greater when compared to the self-trained grammars. This suggests that there is more diversity among the regular grammars than among the self-trained grammars, and explains the greater improvement obtained by the regular product model. It is also important to note that there is more variance among the ST-Prod-Mult grammars, which were trained on disjoint self-labeled training data, and a greater improvement in their product model relative to the ST-Reg and ST-Prod grammars, further supporting the diversity hypothesis. Last but not the least, the trend seems to indicate that the variance of the self-trained grammars would continue increasing if EM training was extended by a few more SM rounds, potentially resulting in even better product models. It is currently impractical to test this due to the dramatic increase in computational requirements for an SM8 product model, and so we leave it for future work.

5.4 Generalization to Broadcast News

We conducted the same set of experiments on the broadcast news data set. While the development set results in Table 6 show similar trends to the WSJ results, the benefits from the combination of self-training and product models appear even more pronounced here. The best single ST-Prod-Mult grammar (89.2 F) alone is able to outperform the product of SM7 regular grammars (88.9 F), and their product achieves another 0.7 absolute improvement, resulting in a significantly better accuracy at 89.9 F.

Model	Rounds	Best	Product
Regular	SM6	87.1	88.6
	SM7	87.1	88.9
ST-Prod	SM6	88.5	89.0
	SM7	89.0	89.6
ST-Prod-Mult	SM6	88.8	89.5
	SM7	**89.2**	**89.9**

Table 6: F-score for various models on the BN development set.

Figure 3 shows again that the benefits of self-training and product models are complementary and can be stacked. As can be observed, the self-trained grammars have increasing F scores as the split-merge rounds increase, while the regular grammars have a slight decrease in F score after round 6. In contrast to the newswire models, it appears that the individual ST-Prod-Mult grammars trained on broadcast news always perform comparably to the product of the regular grammars at all SM rounds, including the product of SM7 regular grammars. This is noteworthy, given that the ST-Prod-Mult grammars are trained on the output of the worse performing product of the SM6 regular grammars. One

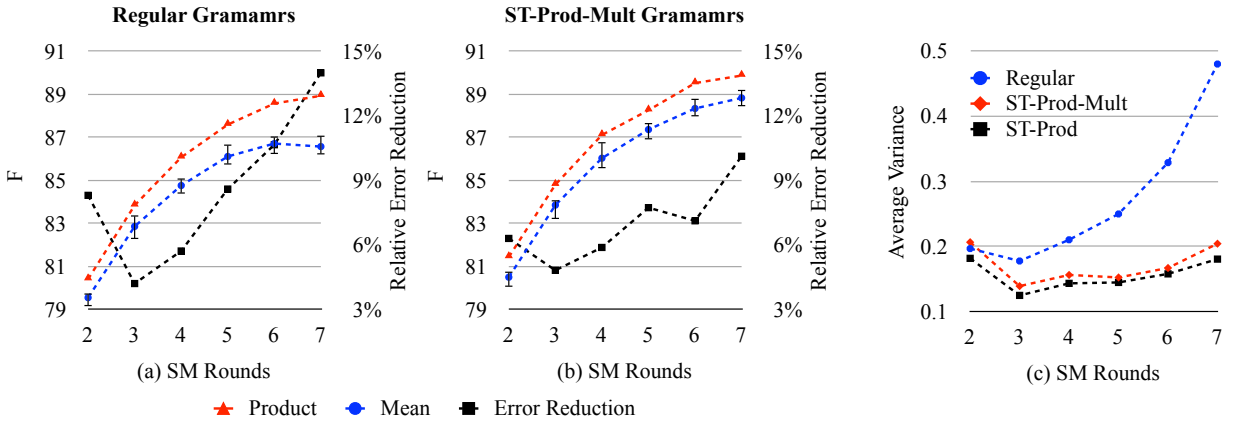


Figure 3: Learning curves of the individual regular (a) and ST-Prod-Mult (b) grammars (average performance, with minimum and maximum values indicated by bars) and their products before and after self-training on the BN development set. The relative error reductions of the products are also reported. (c) The measured average empirical variance among the grammars trained on BN.

possible explanation is that we used more unlabeled data for self-training the broadcast news grammars than for the newswire grammars. The product of the ST-Prod-Mult grammars provides further and significant improvement in F score.

6 Final Results

We evaluated the best single self-trained grammar (SM7 ST-Prod), as well as the product of the SM7 ST-Prod-Mult grammars on the WSJ test set. Table 7 compares these two grammars to a large body of related work grouped into single parsers (SINGLE), discriminative reranking approaches (RE), self-training (SELF), and system combinations (COMBO).

Our best single grammar achieves an accuracy that is only slightly worse (91.6 vs. 91.8 in F score) than the product model in Petrov (2010). This is made possible by self-training on the output of a high quality product model. The higher quality of the automatically parsed data results in a 0.3 point higher final F score (91.6 vs. 91.3) over the self-training results in Huang and Harper (2009), which used a single grammar for parsing the unlabeled data. The product of the self-trained ST-Prod-Mult grammars achieves significantly higher accuracies with an F score of 92.5, a 0.7 improvement over the product model in Petrov (2010).

[8]Our ST-Reg grammars are trained in the same way as in Huang and Harper (2009) except that we keep all unary rules. The reported numbers are from the best single ST-Reg grammar in this work.

Type	Parser		LP	LR	EX
SINGLE	Charniak (2000)		89.9	89.5	37.2
	Petrov and Klein (2007)		90.2	90.1	36.7
	Carreras et al. (2008)		91.4	90.7	-
RE	Charniak and Johnson (2005)		91.8	91.2	44.8
	Huang (2008)		92.2	91.2	43.5
SELF	Huang and Harper (2009)[8]		91.6	91.1	40.4
	McClosky et al. (2006)		92.5	92.1	45.3
COMBO	Petrov (2010)		92.0	91.7	41.9
	Sagae and Lavie (2006)		93.2	91.0	-
	Fossum and Knight (2009)		93.2	91.7	-
	Zhang et al. (2009)		93.3	92.0	-
	This Paper	Best Single	91.8	91.4	40.3
		Best Product	92.7	92.2	43.1

Table 7: Final test set accuracies on WSJ.

Although our models are based on purely generative PCFG grammars, our best product model performs competitively to the self-trained two-step discriminative reranking parser of McClosky et al. (2006), which makes use of many non-local reranking features. Our parser also performs comparably to other system combination approaches (Sagae and Lavie, 2006; Fossum and Knight, 2009; Zhang et al., 2009) with higher recall and lower precision,

but again without using a discriminative reranking step. We expect that replacing the first-step generative parsing model in McClosky et al. (2006) with a product of latent variable grammars would give even higher parsing accuracies.

On the Broadcast News test set, our best performing single and product grammars (bolded in Table 6) obtained F scores of 88.7 and 89.6, respectively. While there is no prior work using our setup, we expect these numbers to set a high baseline.

7 Conclusions and Future Work

We evaluated methods for self-training high accuracy products of latent variable grammars with large amounts of genre-matched data. We demonstrated empirically on newswire and broadcast news genres that very high accuracies can be achieved by training grammars on disjoint sets of automatically labeled data. Two primary factors appear to be determining the efficacy of our self-training approach. First, the accuracy of the model used for parsing the unlabeled data is important for the accuracy of the resulting single self-trained grammars. Second, the diversity of the individual grammars controls the gains that can be obtained by combining multiple grammars into a product model. Our most accurate single grammar achieves an F score of 91.6 on the WSJ test set, rivaling discriminative reranking approaches (Charniak and Johnson, 2005) and products of latent variable grammars (Petrov, 2010), despite being a single generative PCFG. Our most accurate product model achieves an F score of 92.5 without the use of discriminative reranking and comes close to the best known numbers on this test set (Zhang et al., 2009).

In future work, we plan to investigate additional methods for increasing the diversity of our self-trained models. One possibility would be to utilize more unlabeled data or to identify additional ways to bias the models. It would also be interesting to determine whether further increasing the accuracy of the model used for automatically labeling the unlabeled data can enhance performance even more. A simple but computationally expensive way to do this would be to parse the data with an SM7 product model.

Finally, for this work, we always used products of 10 grammars, but we sometimes observed that subsets of these grammars produce even better results on the development set. Finding a way to select grammars from a grammar pool to achieve high performance products is an interesting area of future study.

8 Acknowledgments

This research was supported in part by NSF IIS-0703859. Opinions, findings, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agency or the institutions where the work was completed.

References

Jason Baldridge and Miles Osborne. 2008. Active learning and logarithmic opinion pools for HPSG parse selection. *Natural Language Engineering*.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*.

Kenneth P. Burnham and David R. Anderson. 2002. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. New York: Springer-Verlag.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL*, pages 9–16.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *NAACL*.

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

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson, 2000. *BLLIP 1987-89 WSJ Corpus Release 1*. Linguistic Data Consortium, Philadelphia.

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*.

Denis Filimonov and Mary Harper. 2009. A joint language model with fine-grain syntactic tags. In *EMNLP*, pages 1114–1123, Singapore, August.

Victoria Fossum and Kevin Knight. 2009. Combining constituent parsers. In *NAACL*, pages 253–256.

John Garofolo, Jonathan Fiscus, William Fisher, and David Pallett, 1996. *CSR-IV HUB4*. Linguistic Data Consortium, Philadelphia.

Geoffrey E. Hinton. 1999. Products of experts. In *ICANN*.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *EMNLP*.

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

Percy Liang, Slav Petrov, Michael I. Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *EMNLP*.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor, 1999. *Treebank-3*. Linguistic Data Consortium, Philadelphia.

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*.

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

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

Slav Petrov. 2009. *Coarse-to-Fine Natural Language Processing*. Ph.D. thesis, University of California at Bekeley.

Slav Petrov. 2010. Products of random latent variable grammars. In *HLT-NAACL*.

Brian Roark, Mary Harper, Yang Liu, Robin Stewart, Matthew Lease, Matthew Snover, Izhak Shafran, Bonnie J. Dorr, John Hale, Anna Krasnyanskaya, and Lisa Yung. 2006. SParseval: Evaluation metrics for parsing speech. In *LREC*.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *NAACL*, pages 129–132.

Andrew Smith and Miles Osborne. 2007. Diversity in logarithmic opinion pools. *Lingvisticae Investigationes*.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *ACL*.

Wen Wang and Mary P. Harper. 2002. The superarv language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *EMNLP*, pages 238–247, Philadelphia, July.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, and Ann Houston, 2008. *OntoNotes Release 2.0*. Linguistic Data Consortium, Philadelphia.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *EMNLP*, pages 1552–1560.