Simplicity Bias for Semi-Supervised Parser

Dittaya Wanvarie[†]　Hiroya Takamura[††]　Manabu Okumura[††]
[†]Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology
[††]Precision and Intelligence Laboratory, Tokyo Institute of Technology
dittaya@lr.pi.titech.ac.jp　{takamura,oku}@pi.titech.ac.jp

1 Introduction

Probabilistic context-free grammar (PCFG) is one of the grammars which can be used to parse natural language. PCFG can be straightforwardly extracted from an annotated corpus. Furthermore, parsing with this grammar is very fast. However, PCFG contains not enough information of natural language. For example, the structure such as "*as ... as*" cannot be directly captured.

Another grammar for natural language parsing is tree grammar, which is a generalization of context-free grammar (CFG). In fact, a context-free rule is simply a tree rule with depth 1. Tree grammar can preserve a lot more information than PCFG, but needs more memory to be stored. Moreover, parsing with tree grammar requires longer time than the PCFG parsing.

In both PCFG and tree grammar parsing, there are several possible parses for a sentence. The likelihood criterion can be adopted to select one of these candidate parses. The criterion can be probabilistic function or heuristic. Bod[1] proposed that, for tree grammar parsing, the parse with the shortest tree derivation would be more likely to be correct.

The tree derivation is the number of subtrees, i.e., tree grammar rules, which are combined together to yield the parse. However, there can be several derivations for a specific parse. The shortest derivation is the smallest number of subtrees required to build the parse. According to Zollmann and Sima'an[6], short derivation of tree grammar, i.e., using large subtrees, agrees to the principle of simplicity in the sense of simple derivation. Since small subtrees are included in other larger subtrees, long derivations, i.e., using small subtrees, would be selected only in the case that there is no other shorter derivation.

Instead of parsing with tree grammar from the beginning, we propose to adopt tree grammar parsing to re-parse the result from PCFG parser. The number of derivation required in tree grammar parsing together with the original parser score is used to rerank the n-best parse outputs from the PCFG parser.

We also consider the learning setting in which limits amount of available labeled data. Since annotation of data requires much of human time and labor, the learning should try to acquire necessary information from unannotated data.

Self-training is one technique which can benefit from unlabeled data. The idea is to adapt the supervised model trained by labeled data to more general model using unlabeled data. The algorithm behind is Maximum A Posteriori estimation with Expectation-Maximization algorithm[2].

With only 2000 sentences as labeled data for training, the proposed reranker improves the f-score from 62.89% by PCFG to 64.32%. When 38000 additional unlabeled sentences are provided, the proposed framework achieved the f-score of 65.51% which increases from 63.70% without the reranking process.

2 Related work

Charniak[2] proposed a reranking parser using approximately one million features. Most of the features have binary values. For reranker training process, the author applied the discriminative reranking framework proposed by Collins[3]. The first-stage parser without any reranking is lexicalized PCFG-based which achieved 90.37% of f-score. After the reranking, f-score had been improved to 91.02%.

McClosky et al.[5] proposed a self-training reranking parser based on Charniak's reranking parser. The self-training used the supervised parser to parse unlabeled data. The model is re-trained with original labeled data and parsed unlabeled data. By adding approximately a million sentences of unlabeled data and re-train in self-training framework, the accuracy converged to 92.1%[5].

3 Proposed method

PCFG will be used to generate an n-best parse list. Afterward, the tree grammar parsing is applied to rerank the parses in the list. The tree derivation count of each parse together with the original parse score from PCFG is used as features to rerank the parse.

The tree derivation counting process is the same as the tree grammar parsing. Goodman[4] has proposed the PCFG reduction of tree grammar with the shortest derivation criterion which provides fast dynamic programming algorithm for tree grammar parsing. In order to apply the tree grammar parsing to tree derivation counting, we strictly considers only the tree rules in the parse output from PCFG. By this method, the size of tree grammar rules involved in the parsing is reduced.

The tree derivation score used in this work is the difference of tree-derivation count from the minimum tree-derivation count in that parse list. For example, if the parse list contains parses A, B, C whose derivation counts are 5, 6, 8, respectively. The minimum derivation count is 5. Thus, the tree score of each parse would be 0, 1, and 3, respectively.

4 Experiment

4.1 Data setting

Sections 2 to 21 from conventional Wall Street Journal corpus are used as training, sections 0, 1, 21 as development, and section 23 as test data respectively. All punctuations are removed. Following, WSJ2000, WSJ10000 represent the settings which use the first 2000 sentences and the first 10000 sentences of the training set as labeled data respectively. WSJ2-21 represents the whole training set. Unlabeled data are the rest of data in the training set.

4.2 training setting

The proposed reranker is evaluated in 2 modes, the supervised and semi-supervised modes. The supervised setting is intended to measure the performance of the reranker without any help from unlabeled data. The semi-supervised setting is explored to judge the improvement due to unlabeled data.

We use the same self-training setting as stated in McClosky et al.[5]. However, this work is intended to evaluate the ability of the proposed reranker when very small size of labeled data are available. Only limited number of labeled data are used, leaving the rest as unlabeled.

4.3 Selection function

In order to rank a parse using the 2 proposed features, we proposed the heuristic naiveBest selection, which will rank parses by the tree score in ascending order. If there are several parses with the same tree score, these parses are ranked by their original ranks from the parser.

	WSJ0	WSJ1	WSJ22
parserBest	63.6	63.81	62.59
naiveBest	**64.85**	**64.68**	**64.11**
oracle	78.21	78.28	77.84

Table 1: Selecting function result on WSJ2000

The parserBest selection, which uses only the original rank from the PCFG parser, is applied as a baseline. At the same time, the oracle selection which always selects the best parse from the list, is the upper bound reranker.

4.4 Result and discussion

Table 1 shows the f-score of each selection function in supervised settings. Using only small amount of labeled data, WSJ2000, the naive selection obtains 1.21% accuracy improvement on average from the first-stage PCFG parser.

In semi-supervised setting with the parserBest reranker, as shown in Figure 1(a), the accuracy increases approximately 0.57% with WSJ2000+rest in the first iteration, and continues decreasing through other next iterations. However, with more labeled data and less unlabeled data, the accuracy with WSJ10000+rest does not increase even at the first iteration.

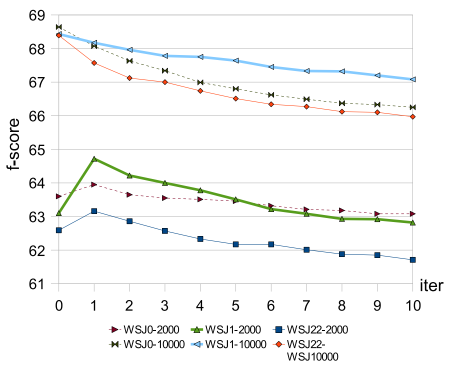
The improvement of accuracy with WSJ2000+rest of the parserBest reranker at the first iteration relies on recall improvement which is statistically significant with α=0.05%.

From the first to the fourth iteration, the decreases of accuracies in WSJ1 and WSJ22 rely on the drop of precision with α=1%.

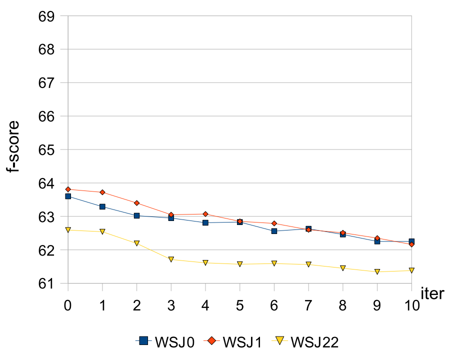
In contrast, the accuracy of the parserBest model when training with WSJ2000 as labeled data and retraining with smaller, 6000 unlabeled sentences, decreases even at the first iteration as shown in Figure 1(b). The trend of the result is the same as in the experiment of WSJ10000+rest. The decrease after adding unlabeled data due to the drop of precision is significant with α=0.01% in the first iteration, and with α=1% from the first to the third iteration. Changes in other iterations are not significant which indicates the convergence.

Figure 2(a) shows that the accuracy of naiveBest reranker increases and converges to a constant level. The improvement relies on the uprise of recall as in the parserBest reranker, with α=0.01%. The difference of accuracy in other next iterations is not significant with α=5%, which also indicates the convergence.

The naiveBest reranker also improves the accuracy when a small amount of unlabeled data is provided. The increase is due to an improvement of recall with

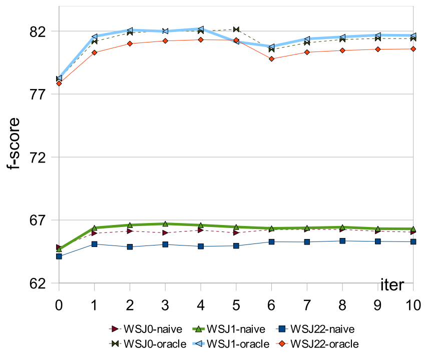


(a) WSJ2000+rest

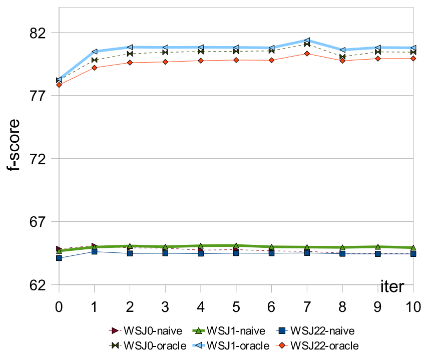


(b) WSJ2000+6000

Figure 1: Result with WSJ2000+rest and WSJ2000+6000 on WSJ0, WSJ1, WSJ22 using parserBest selection



(a) WSJ2000+rest



(b) WSJ2000+6000

Figure 2: Result on WSJ0, WSJ1, WSJ22 with WSJ2000+rest and WSJ2000+6000 using each selection function

	Model	f-score
Supervised	WSJ2000	62.89
	SVR-RBF	64.24
	WSJ2000-naive	64.32
Semi-supervised	parserBest	63.70
	naiveBest(7^{th})	**65.51**
	oracle	81.43

Table 2: Result of each selection function with WSJ2000+rest on WSJ23

α=1%. Figure 2(b) shows that the proposed naiveBest reranking function also converges with this dataset.

With WSJ2000 and the rest as unlabeled, though any iterations from the second is not statistically different, the seventh iteration of naiveBest reranking parser achieves the best f-score of 65.96% in average of the three development sets. Hence, the model of this iteration is employed as the final model for naiveBest reranker. The comparison of result among selection functions is stated in Table 2.

The first two columns in Table 2 are results of supervised parsers. Applying the naiveBest reranker improves the accuracy from 62.89% to 64.32%. By adding unlabeled data with self-training, the proposed reranker achieved the f-score of 65.51%, which is higher than all of the supervised parsers in the experiments.

In Figure 1(a), the semi-supervised setting of parserBest selection improves the accuracy in WSJ2000+rest only at the first iteration. In other following iterations, the accuracy continues decreasing.

But in Figure 1(b), the parserBest selection on the same WSJ2000 with fewer unlabeled data suffers from a decrease of accuracy even at the first iteration. The result is the same as one from WSJ10000+rest dataset

	parserBest	naiveBest
1	76.39	46.36
2	31.54	24.65
3	20.57	14.85
4	13.9	10.5
5	9.92	7.71
6	7.31	5.82
7	5.88	4.55
8	5.01	3.6
9	4.1	3.12

Table 3: Percent of different parses of unlabeled data over iteration with WSJ2000+rest

in Figure 1(a). Note that the ratio between labeled and unlabeled data is approximately the same in these two cases.

A large amount of unlabeled data will adjust probabilities of grammar rules in the first-stage PCFG parser to more appropriate values using Expectation-Maximization algorithm. As the first-stage parser contains insufficient information, adding unlabeled data without further information may lead to the worse accuracy. In contrast, the naiveBest reranker predicted score by considering both information from PCFG and from the tree grammar. As a result, the naiveBest reranker is supposed to be more accurate than the parserBest to predict the correct parse.

From Figures 2(a) and 2(b), the naiveBest reranker indicates the convergence over iterations while the parserBest continues decreasing. The convergence relies on the number of changed sentences during iterations.

According to Table 3, the number of changed training sentences is reduced over the training iterations. This reduction indicates the convergence. The reduction of changed sentences is an effect from Expectation-maximization algorithm used in self-training setting. During iterations, the parses which are mostly fit to the model are chosen, then leading to convergence.

The naiveBest selection rapidly reduces the number of changed-parsed-unlabeled sentences. One reason is that the same parse always has the same tree-derivation count in every iteration. In the naive reranker, only parses with the best tree score would be considered. On the other hand, the parserBest reranker considers all parses in a parse list at a time. Therefore, there is less chance that parserBest method will assign the same parse to the same rank as assigned in the previous iteration.

The parserBest selection also signifies a convergence as the number of different sentences also decreases, but with slower rate than the naiveBest selection.

5 Conclusion

In this work, the shortest tree derivation count is applied as a feature to rerank parses from n-best list generated by a PCFG parser. The heuristic naiveBest is employed as a selection function. The performance of the proposed function is evaluated in both supervised and semi-supervised settings. In superivsed learning, the proposed reranker achieves higher accuracy than the parserBest baseline.

In semi-supervised setting, the naiveBest reranking parser converges faster than the parserBest reranking parser. Since the naiveBest selection assigns the same score to the same parse in every iteration, it is more likely that the same parse will be re-selected in subsequent iterations, thus converging faster. The experiment also proved that the use of unlabeled data is effective in improving the accuracy of the model. Even with small amount of unlabeled data, the proposed reranker can improve the accuracy of the first-stage PCFG parser.

6 Future work

The proposed naiveBest selection function is only one of several possible ways to score parses. Other possible functions such as linear regression, support vector regression (SVR), etc. can also be used as a selection function. In this experiment, the support vector regression has been evaluated. It provided an encouraging result. However, these parameters for SVR needs to be optimized.

The proposed method can also be able to be applied to any parsers which can produce n-best parse outputs. However, the current work is only evaluated on PCFG parser. The state-of-the-art parsers such as Charniak parser, Collins parser are also able to produce n-best outputs. There should be further evaluations on these parsers.

References

[1] R. Bod. Parsing with the shortest derivation. In *COLING*, pages 69–75, 2000.

[2] E. Charniak and M. Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*, pages 173–180, 2005.

[3] M. Collins. Discriminative reranking for natural language parsing. In *Proc. 17th International Conf. on Machine Learning*, pages 175–182. Morgan Kaufmann, San Francisco, CA, 2000.

[4] J. Goodman. *Data oriented parsing*, chapter 8, pages 125–146. CSLI, 1992.

[5] D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *HLT-NAACL*, pages 152–159. Association for Computational Linguistics, 2006.

[6] A. Zollmann and K. Sima'an. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, 10(2/3):367–388, 2005.