Multiple TreeBanks Integration for Chinese Phrase Structure Grammar Parsing Using Bagging

Meishan Zhang   Wanxiang Che   Ting Liu
School of Computer Science and Technology
Harbin Institute of Technology, Harbin, China
{mszhang, car, tliu}@ir.hit.edu.cn

Abstract

We describe our method of traditional Phrase Structure Grammar (PSG) parsing in CIPS-Bakeoff2012 Task3. First, bagging is proposed to enhance the baseline performance of PSG parsing. Then we suggest exploiting another TreeBank (CTB7.0) to improve the performance further. Experimental results on the development data set demonstrate that bagging can boost the baseline F1 score from 81.33% to 84.41%. After exploiting the data of CTB7.0, the F1 score reaches 85.03%. Our final results on the official test data set show that the baseline closed system using bagging gets the F1 score of 80.17%. It outperforms the best closed system by nearly 4% which uses a single model. After exploiting the CTB7.0 data, the F1 score reaches 81.16%, demonstrating further increases of about 1%.

1 Introduction

Over the past decade, Phrase Structure Grammar (PSG) parsing has been investigated by many researchers. Most methods of PSG parsing exploited some manly annotated corpus and proposed a single statistical model (Petrov and Klein, 2007; Zhang and Clark, 2009) based on the corpus. For Chinese, Tsinghua Chinese Treebank (TCT) (Qiang, 2004) and Penn Chinese TreeBank (CTB) (Xue et al., 2005) are two most popular manly annotated corpus.

In this paper, we are especially interested in parser combination. Many past works have suggest a number of methods for parser combination. These methods concern on combing different parsers which are trained on the same corpus. Sagae and Lavie (2006) proposed a constituent reparsing method for multiple parsers combination. Zhang et al. (2009) proposed a linear model-based general framework to combine several lexicalized parsers (Collins, 1999; Zhang and Clark, 2009) and un-lexicalized parsers (Petrov et al., 2006; Petrov and Klein, 2007).

Out method is different from the past works in that we combine different parsers which exploit the same method but the models of which are trained on different corpus. We adopt Berkeley parser[1] (Petrov et al., 2006; Petrov and Klein, 2007) to train our sub-models. It is an un-lexicalized probabilistic context free grammar (PCFG) parser. At the beginning, we train a number of submodels by sampling TCT corpus repeatedly, and meanwhile train a number of submodels by sampling CTB corpus repeatedly. Then we combine these submodels by reparsing the parsing results of them using the CKY-parsing algorithm (Song et al., 2008).

To enable using CKY-parsing algorithm for combining, we must handle the following two issues:

1. Binarization should be applied to the parsing results of submodels.

2. The grammars of TCT corpus are very different that of CTB corpus. We should transform CTB grammars into TCT grammars before final combination.

If these two issues have been done already, we can apply CKY reparsing algorithm and get the final parsing result.

The rest of the paper is organized as follows. Section 2 introduces the overall system architecture. And then we introduce our method in detail. In section 3 we present the binarization algorithm used in the system. Section 4 describes the CKY reparsing algorithm. Section 5 describes our baseline method and multiple TreeBank bagging

[1]http://code.google.com/p/berkeleyparser

method systematically. Section 6 shows the experimental results and finally in section 7 we conclude our method and give our future works.

2 System Architecture

During the training phase, we sample the training corpus of TCT and CTB repeatedly, exploiting these sampled corpus to train a number of submodels. In the test phase, first we parse a sentence using these submodels, and then binarize the parsing results, extracting the binarized grammars together with their weights, and finally exploit CKY reparsing algorithm to get our final parsing results according to the weighted grammars . For the CTB results, we should add an extra transformation process to map the CTB grammars to TCT grammars. The transformation model are trained by mapping gold TCT results and Figure 1 shows the architecture of the training and testing process.

3 Binarization

The binarization process aims at a better combination using CKY reparsing. We must ensure that the binarization process is reversible.

For the unary grammar, we simply merge the label of leaf node into its parent node. We add a special mark during the merging so that we can reverse the merging conveniently.

For the grammars whose arity are more than two, we don't use a simple left most binarization or right most binarization algorithm. As these simple binarization can make the mapping between different TreeBanks very complex. Our goal is to get a better understanding binarization results which the grammars extracted from the different TreeBanks can be more easily forming one-to-one mapping. The most popular binary grammars extracted from the TreeBank are exploited for binarization. By this method, the grammars of binarization can be mostly understood.

We describe our binarization algorithm to handle the high-arity grammars. To prepare for binarization, we need collect binarization grammar and their weights. We denote the collection results by $\mathrm{G_{bin}} = \{(A \rightarrow BC, \mathrm{freq})\}$. This process is done simply extracting all the binary grammars from the original TreeBank and assigning the corresponding weight by their appearance frequency. The pseudo-code of the binarization is shown in Algorithm 1. We can get the binarization tree of a PS structure by applying Algorithm 1 on each non-terminal node from up to bottom.

The TCT training corpus has been already binarized that it contains unary and binary grammars, thus we can get the binarization results for the output of TCT submodels by simply merging unary grammars. The CTB corpus contains grammars of variety number of arity. We need first merge the unary grammars and then apply algorithm 1 to get the binarization results.

4 CKY Parsing

In this section, we describe the CKY parsing algorithm which aims for bagging system. The form of rules used CKY parsing are defined by a tuple $(A \rightarrow BC, \mathrm{s}, \mathrm{m}, \mathrm{e})$. It denotes a binary tree structure, $A \rightarrow BC$, the start position is s, middle position is m which is also the end of tree labeled by B, and the end position e. The rules and their weights are basic input grammar for CKY parsing, and we denote it by $\mathrm{G_{cky}} = \{((A \rightarrow BC, s, m, e), \mathrm{w})\}$. The pseudo-code of the CKY parsing is shown in Algorithm 2. The algorithm is very similar to the binarization algorithm.

5 Methods

5.1 Baseline Bagging System

The training process of the baseline bagging system:

1. Sample k new training corpus from the overall TCT corpus. Assuming the size of overall TCT corpus is n, we repeatedly sample the overall TCT corpus for k times. Each time we get a new training corpus whose size is $64.3\% \times n$.

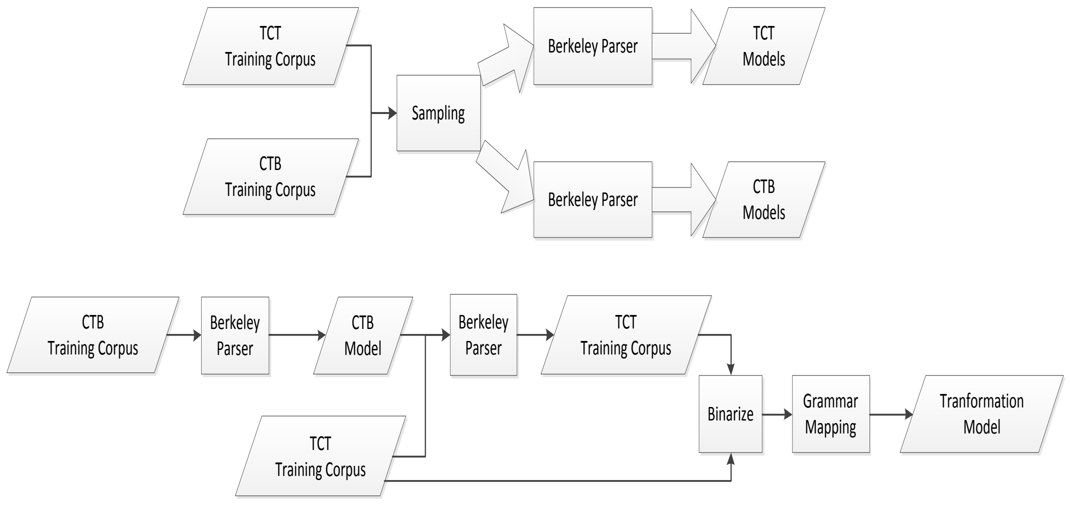
2. Train k submodels using the sampled k new train corpus.

The decoding process of the baseline bagging system:

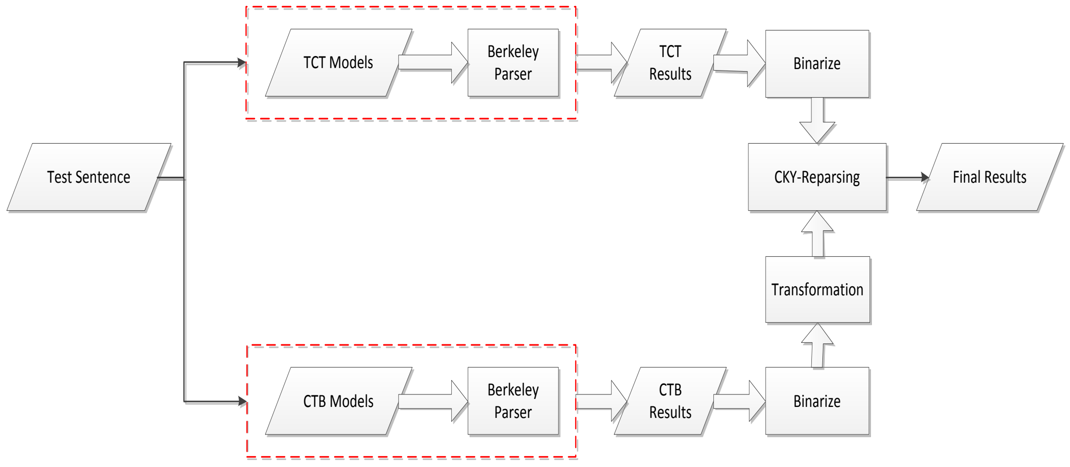
1. Parse the input sentence by the k submodels and get k PS results of the sentence.

2. Binarize the k PS results.

3. Generate the grammar $\mathrm{G_{cky}}$. We extract all rules $(A \rightarrow BC, s, m, e)$ from the k PS results. The weight of each rule equals their frequency.



(a) Train



(b) Test

Figure 1: System architecture.

Algorithm 1 Binarization Algorithm. L denotes the set of non-terminal labels, and label(tr) denote the root label of tree tr.

```
Input: $\mathrm{G_{bin}}$, Tree : $\mathrm{tr}_0 \longrightarrow \mathrm{tr}_1 \cdots \mathrm{tr}_n$
Initialization:
  for all $i \in \{1 \cdots n\}$, for all $A \in L$
    if $\mathrm{label}(\mathrm{tr}_i) = A$, $\pi(i, i, A) = 1$
    else $\pi(i, i, A) = 0$
Compute:
  for all $d \in \{1 \cdots n-1\}$
    for all $i \in \{1 \cdots n-j\}$
      set $j = i + l$
      for all $A \in L$
        $\pi(i, j, A) = \max_{A \to BC \in \mathrm{G_{bin}}, i<s<j} \pi(i, s, B) + \pi(s+1, j, C) + \mathrm{G_{bin}}(A \to BC)$
        $\delta(i, j, A) = \arg\max_{A \to BC \in \mathrm{G_{bin}}, i<s<j} \pi(i, s, B) + \pi(s+1, j, C) + \mathrm{G_{bin}}(A \to BC)$
Create a new tree tr:
  From $\delta(1, n, \mathrm{label}(\mathrm{tr}_0))$, generate middle nodes recursively.
  Add a special mark to the label of all middle nodes, which are used to restore.
Return: Binarized tree tr
```

Algorithm 2 CKY Parsing Algorithm. T denote the set of POS tags.

```
**Input:** $G_{\text{cky}}$, leaves : $\text{tr}_1 \cdots \text{tr}_n$
**Initialization:**
  for all $i \in \{1 \cdots n\}$, for all $t \in T$
    if label($\text{tr}_i$) $= t$, $\pi(i, i, A) = 1$
    else $\pi(i, i, A) = 0$
**Compute:**
  for all $d \in \{1 \cdots n - 1\}$
    for all $i \in \{1 \cdots n - j\}$
      set $j = i + l$
      for all $A \in L$
        $\pi(i, j, A) = \max_{(A \to BC, i, s, j) \in G_{\text{cky}}} \pi(i, s, B) + \pi(s + 1, j, C) + G_{\text{bin}}(A \to BC, i, s, j)$
        $\delta(i, j, A) = \arg\max_{(A \to BC, i, s, j) \in G_{\text{cky}}} \pi(i, s, B) + \pi(s + 1, j, C) + G_{\text{bin}}(A \to BC, i, s, j)$
**Create a new tree tr:**
  From $\delta(1, n, \text{root})$, generate middle nodes recursively.
**Return the tree tr**
```

4. Generate the leaves : $\text{tr}_1 \cdots \text{tr}_n$. Each leaf tr_i are composed by a word w_i and its POS tag t_i, forming $t_i \to w_i$. As each word can have k results, thus we can use voting to assign the word's best POS tag t_i.

5. Reparse the sentence using CKY parsing algorithm with G_{cky} and leaves : $\text{tr}_1 \cdots \text{tr}_n$.

5.2 Bagging System Exploiting CTB Corpus

The training process of the baseline bagging system:

1. Sample k new training corpus from the overall TCT corpus and sample k new training corpus from the overall CTB corpus. We will get $2k$ new training corpus in this step.

2. Train $2k$ submodels using the sampled $2k$ new train corpus, where k submodels are the TCT stype parsers and the other k submodels are the CTB style parsers.

3. Train a transformation model from CTB style to TCT style $\text{Map}_{\text{CTB}\to\text{TCT}}$. It can be finished by the following steps.

   (a) Train a model using all CTB Corpus,
   (b) Parse the entire TCT training corpus,
   (c) Binarize the gold TCT style PS structure,
   (d) Binarize the predicted CTB style PS structure,
   (e) Compare the gold TCT results and the predicted results and get a final transformation model.

   For a TCT grammar $(A_{\text{tct}} \to B_{\text{tct}}C_{\text{tct}}, s_{\text{tct}}, m_{\text{tct}}, e_{\text{tct}})$ and a CTB grammar $(A_{\text{ctb}} \to B_{\text{ctb}}C_{\text{ctb}}, s_{\text{ctb}}, m_{\text{ctb}}, e_{\text{ctb}})$, if $(s_{\text{tct}}, m_{\text{tct}}, e_{\text{tct}}) = (s_{\text{ctb}}, m_{\text{ctb}}, e_{\text{ctb}})$, we would add a mapping rule $(A_{\text{tct}} \to B_{\text{tct}}C_{\text{tct}}, A_{\text{ctb}} \to B_{\text{ctb}}C_{\text{ctb}}, s_{\text{tct}}, m_{\text{tct}}, e_{\text{tct}})$ to $\text{Map}_{\text{CTB}\to\text{TCT}}$, and if $(s_{\text{tct}}, e_{\text{tct}}) = (s_{\text{ctb}}, e_{\text{ctb}})$, we would add a mapping rule $(A_{\text{tct}} \to B_{\text{tct}}C_{\text{tct}}, A_{\text{ctb}} \to B_{\text{ctb}}C_{\text{ctb}}, s_{\text{tct}}, e_{\text{tct}})$ to $\text{Map}_{\text{CTB}\to\text{TCT}}$.

The decoding process of the baseline bagging system:

1. Parse the input sentence by the k TCT submodels and get k PS results of TCT style.

2. Binarize the k PS results.

3. Generate the grammar G_{cky}. We extract all rules $(A \to BC, s, m, e)$ from the k PS results. The weight of each rule equals their frequency.

4. Generate the leaves : $\text{tr}_1 \cdots \text{tr}_n$. Each leaf tr_i are composed by a word w_i and its POS tag t_i, forming $t_i \to w_i$. As each word can have k results, thus we can use voting to assign the word's best POS tag t_i.

5. Parse the input sentence by the k CTB submodels and get k PS results of CTB style.

6. Adjust the grammar G_{cky} by k PS results of CTB style. First we extract all grammars from the k PS results. For each grammar $(A_{\text{ctb}} \to B_{\text{ctb}}C_{\text{ctb}}, s_{\text{ctb}}, m_{\text{ctb}}, e_{\text{ctb}})$, we find

its mapping rule from $\text{Map}_{\text{CTB}\rightarrow\text{TCT}}$. The mapping rule result can be either $(A_{\text{tct}} \rightarrow B_{\text{tct}}C_{\text{tct}}, A_{\text{ctb}} \rightarrow B_{\text{ctb}}C_{\text{ctb}}, s_{\text{tct}}, m_{\text{tct}}, e_{\text{tct}})$ or $(A_{\text{tct}} \rightarrow B_{\text{tct}}C_{\text{tct}}, A_{\text{ctb}} \rightarrow B_{\text{ctb}}C_{\text{ctb}}, s_{\text{tct}}, e_{\text{tct}})$. Then we traverse all grammars in G_{cky}, if the grammar matches with $(A_{\text{tct}} \rightarrow B_{\text{tct}}C_{\text{tct}}, s_{\text{tct}}, m_{\text{tct}}, e_{\text{tct}})$ or partially matches with $(A_{\text{tct}} \rightarrow B_{\text{tct}}C_{\text{tct}}, s_{\text{tct}}, e_{\text{tct}})$, then its weight will be increased by value α. The value α should be adjusted according to development set.

7. Reparse the sentence using CKY parsing algorithm with G_{cky} and leaves : $\text{tr}_1 \cdots \text{tr}_n$.

6 Experiments

6.1 Data Set

The task organizers have offered 17,758 annotated sentences for train our model. They are chosen from TCT corpus. Before they share us for train, the trees which have more than two leaves have been processed to ensure all the grammars in the train sentences containing only unaries and binaries. We use the training section of CTB7.0 to to train the models of CTB. The training sections are selected by the documents of LDC2010T07. The total number of CTB training is 46,572. To adjust some parameters in our model, we split a development data set from the entire training corpus. After get the value of these parameters, we retrain our system using all the corpus. Table 1 shows the statistics of the data set.

Corpus	Section	# sent.
Parameter Adjusting	Train	15802
	Devel	1756
CTB7.0	Train	46572
Final Test	Train	17558
	Test	1000

Table 1: Statistics of Data Set.

6.2 Parameter Adjusting

First we look at how bagging numbers k influence the the baseline bagging system. In this work, we set the bagging num $k = 15$. Figure 2 displays the result. As is shown in Figure 2, the performance increments gradually when the bagging number becomes larger. The performance is better than a single model since the bagging number is 3.

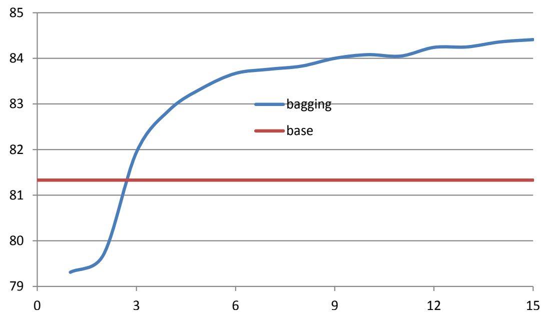


Figure 2: Bagging results. The baseline denotes the model which doesn't exploit sampling and bagging.

Second we adjust the parameter α by development also. The α should be less than 1 by intuition. We gradually increase the value of α from 0.5 to 1.0. Figure 3 display the results on development set. According to the results, we set $\alpha = 0.9$

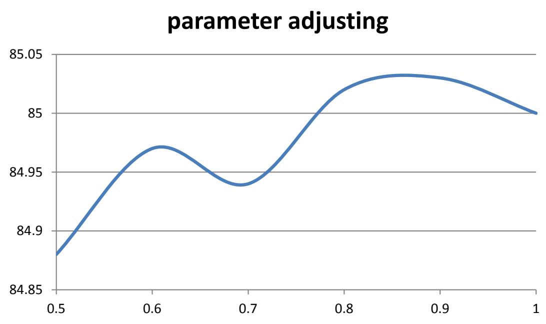


Figure 3: Parameter adjusting result.

6.3 Final Results

First, to get a better understanding of our system, we show the results on the development data set. **Berkeley** denotes the result of Berkeley parser which doesn't use bagging. **Bbag** denotes our baseline bagging system which uses only TCT corpus. **Cbag** denotes our final system which uses both TCT corpus and CTB corpus. Table 2 displays the results.

System	P	R	F1
Cbag	**85.04**	**85.03**	**85.03**
Bbag	84.4	84.42	84.41
Berkeley	81.31	81.35	81.33

Table 2: Final results on the development set.

Table 3 displays our final result on test data which the task organizers offered. **BestClosedSingle** denotes the best closed system of the task.

From the results in both Table 2 and Table 3, we can find that bagging is a very simple and effective method to combine multiple TreeBanks.

System	P	R	F1
Cbag	**81.20**	**81.12**	**81.16**
Bbag	80.23	80.11	80.17
BestClosedSingle	76.35	76.20	76.27

Table 3: Final results on the development set.

7 Conclusions and Future Work

In this paper, we propose to exploit bagging to enhance the performance PSG parsing. The method is very simple and effective. The bagging is implemented upon a CKY reparsing algorithm. We introduce CKY reparsing algorithm in detail and introduce the preprocess binarization algorithm. By bagging, we can achieve increases nearly 3% in F1 score. Further, we exploit bagging to integrate CTB corpus to enhance PSG parsing. And finally, we have achieved further increases nearly 1% after using CTB7.0.

In the future, we will investigate the transformation methods to better integrate multiple TreeBanks. We are very interested in statistical models to finish this transformation.

Acknowledgments

This work was supported by National Natural Science Foundation of China (NSFC) via grant 61133012, the National 863 Major Projects via grant 2011AA01A207, and the National 863 Leading Technology Research Project via grant 2012AA011102.

References

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Pennsylvania University.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

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

Zhou Qiang. 2004. Annotation scheme for chinese treebank. *Journal of Chinese Information Processing*, 18(4):1–8.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA, June. Association for Computational Linguistics.

Xinying Song, Shilin Ding, and Chin-Yew Lin. 2008. Better binarization for the CKY parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 167–176, Honolulu, Hawaii, October. Association for Computational Linguistics.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171, Paris, France, October. Association for Computational Linguistics.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1552–1560, Singapore, August. Association for Computational Linguistics.