HIT Dependency Parsing : Bootstrap Aggregating Heterogeneous Parsers

Meishan Zhang, Wanxiang Che, Yijia Liu, Zhenghua Li, Ting Liu
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{mszhang, car, yjliu, lzh, tliu}@ir.hit.edu.cn

Abstract

The paper describes our system of *Shared Task on Parsing the Web*. We only participate in dependency parsing task. A number of methods have been developed for dependency parsing. Each of the methods adopts very different view of dependency parsing, and each view can have its strengths and limitations. Thus system combination can have great potential to further improve the performance of dependency parsing. In this work, *Bootstrap Aggregating (Bagging)* is chosen to combine these methods. This approach obtains significantly improvements for dependency parsing, and especially we achieves a UAS of 93.88%, LAS of 91.88% on WSJ domain, which is the top result of all participated systems. We tried to use unlabeled data offered by this task as well, and unfortunately we received little improvements through tri-training. Finally, our final bagging system ranked thirdly of the shared task.

1 Introduction

Past research works of dependency parsing always employ a fixed corpus to train and test, achieving accuracies above 90%. However, the accuracies will drop drastically when applied to web corpus, which makes web applications that rely on parsing, such as machine translation, sentiment analysis and information extraction suffer a lot. *Shared Task on Parsing the Web* (Petrov and McDonald, 2012) is designed to make researchers follow web parsing more conveniently.

Dependency parsing has been studied intensively for several years. Many methods have been proposed for it, such as graph-based, transition-based and constituent-based methods. Graph-based methods factor a dependency tree into arcs, and parsing is performed by searching for the highest scoring tree. Transition-based methods build a dependency tree through a sequence of shift-reduce operations, given previous decisions and current state, parsing is performed by greedily choosing the highest scoring operate out of each successive parsing state. Constituent-based methods make uses of a constituent parser, outputting dependency results by converting phrase-structures to dependency structures using a set of rules. Since graph-based and transition-based methods do not need any formal grammars whereas constituent-based methods are based on context-free grammar, thus we can call the former two kinds as grammar-free methods and the last as grammar-based methods. All these approaches adopt very different views of dependency parsing. Each view has its strengths and limitations. A natural idea is to combine these methods together.

Sun (2012) systematically study applying *Bootstrap Aggregating (Bagging)* to combine heterogeneous views in Chinese language processing. In particular, syntax-free and syntax-based models are combined to achieve more accurate POS tagging; grammar-free and grammar-based models are combined to obtain better dependency parsing results. This research inspires our work here.

In this work, we exploit bagging to combine three parsers together: Mate parser[1] (Bohnet, 2010) which

[1] http://code.google.com/p/Mate-tools/

is a graph-based method, Jun parser which is a transition-based parser[2] (Hatori et al., 2011) and Berkeley parser[3] (Petrov et al., 2006) which is a constituent parser.

The POS tags should not be neglected as they are always indispensable components for dependency parsing. In general, the performance of dependency parsing can be improved when the accuracy of input POS tagging increases. In our system, we employ *stacking* to combine the fine-grained POS tagging results produced by a CRF Tagger, fine-grained POS tagging results produced by constituent parser and the coarse-grained POS tagging results produced by a CRF tagger together.

We tried to utilize unlabeled corpus to further improve the performance of our bagging system as well. Tri-training (Ming and Zhou, 2005) was applied to generate automated training corpus for Mate parser. Although the performance of Mate parser has an improvement of 0.88%, the performance of our bagging system receives little improvements.

In the following of this paper, we will describe our system in detail, and report several experimental results.

2 System Description

2.1 POS Tagging

POS tags are essential inputs for dependency parsing. The accuracy of POS tagging can influence the performance of dependency a lot. Normally, if the accuracy of POS tagging increases, the UAS and LAS of dependency will increase as well.

Figure 1 shows the flow chart of our POS tagging. Firstly, we obtain two POS tagging results for each sentence: fine-grained POS tags(t^b) produced by Berkeley parser and coarse-grained POS tags(t^c) produced by a CRF tagger (CRF_{coarse}). Next, these two POS tagging results are used to produce guiding features in our final CRF tagger ($CRF_{fine-guide}$) which outputs fine-grained POS tags. Table 1 lists the features employed in our CRF tagger, where w denotes a word and t denotes the output POS tag. CRF_{coarse} only uses basic features and $CRF_{fine-guide}$ uses both features. The lexicon is extracted from the training corpus when the frequency of pair (w, t) is higher than 5.

Basic Features
$w_i, w_iw_{i+1}, w_iw_{i-1}, w_iw_{i+2}, w_iw_{i-2}, w_{i-1}w_iw_{i+1}$
Prefix of $w_i = P$, Suffix of $w_i = S$, where $\parallel P, S \parallel < 5$
whether w_i contains a digit, capitalized
Inlexicon(w_0, t), for any possible POS tag t
Guide Features
$t_i^b, t_i^b t_{i+1}^b, t_i^b t_{i-1}^b, t_i^c, t_i^c t_{i+1}^c, t_i^c t_{i-1}^c, t_i^b t_i^c$

Table 1: Feature templates of our CRF tagger.

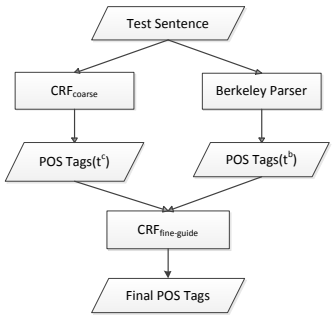

Figure 1: Flow chart of POS tagging.

During training phase, we use the tool of CRFSuite[4] to learn parameters of CRF Tagger, and the tool of Berkeley parser to train constituent model. To generate the training corpus for $CRF_{fine-guide}$, t^b and t^c are produced via five-fold cross-validation.

2.2 Dependency Parsing

Figure 2 shows the overall framework of our dependency parsing. The box with red dashed line demonstrates that all the elements inside the box compose an ensemble system, the fine arrow means that a single result is produced by the source, and the thick arrow means that multiple results are produced by the source.

The training set of Mate parser and Jun parser is WSJ training corpus of dependency structure with auto POS tags which are generated through five-fold cross-validation by $CRF_{fine-guide}$, and Berkeley parser uses WSJ training corpus of constituent

[2]We thank Jun Hatori very much for the code of transition parser shared for us. The pipeline method is exploited in our system, and we can refer to Zhang and Nivre (2011) for details of the parser.

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

[4]http://www.chokkan.org/software/crfsuite/

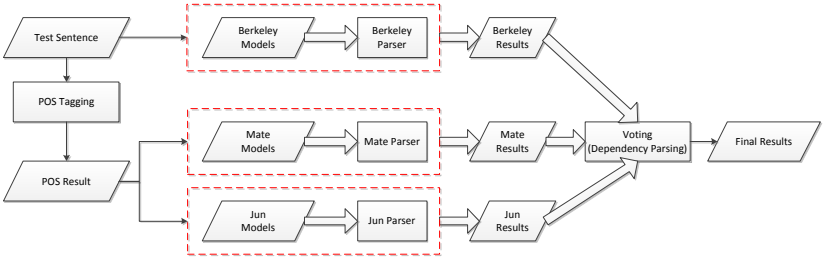

Figure 2: Workflow of our system.

structure to train models. In the training phase, assuming the training set D of size n for Mate parser, m new training sets D_i of size $61.8\% \times n$ are generated by sampling without replacement. Each D_i is separately used to train a Mate model. Thus we can get m models for Mate parser. Similarly we can get m models for Berkeley parser and m models for Jun parser. When decoding, first we can get m results by Berkeley parser, m results by Mate parser and m results by Jun parser, and then combine these results via word-by-word voting.

3 Experiments

3.1 POS Tagging

Table 2 shows how the accuracy of POS tagging influences the result of dependency parsing on weblogs developing set. The dependency parsing model was kept unchanged, and it was trained by Mate with automated POS tags. We compare following POS taggers: MXPOST (Ratnaparkhi, 1996), Stanford POS Tagger (Toutanova et al., 2003), Berkeley parser, CRF POS tagger(only basic features are used), $\text{CRF}_{\text{fine-guide}}$ and GOLD. As is shown in Table 2, the performance of dependency parsing improves as the accuracy of POS tagging increases except one case.

POS tagger	Pos	UAS	LAS
MXPOST	93.52	88.04	84.67
Stanford POS tagger	94.14	88.43	85.27
Berkeley parser	94.24	89.24	86.08
CRF POS tagger	94.61	88.92	85.86
$\text{CRF}_{\text{fine-guide}}$	95.04	89.42	86.42
Gold	100	90.95	88.96

Table 2: Relation between POS tag and dependency parsing.

Table 7 displays the final POS tagging accuracies on the six domains' testing section.

3.2 Dependency Parsing

Firstly, we show the performance of the bagging on WSJ developing set in Table 3. The bagging times m is set to nine here. *baseline* denotes the model trained on the whole WSJ training corpus. We can see that bagging is very effective to improve the performance of dependency parsing. Secondly,

	method	UAS	LAS
Berkeley	baseline	92.33	89.90
	bagging	93.44	91.17
Mate	baseline	92.59	90.27
	bagging	92.90	90.62
Jun	baseline	91.53	–
	bagging	91.88	–

Table 3: Performance of bagging models on WSJ developing set.

we show the performance of the multi-view bagging model on WSJ developing set in Table 4. Multi-view bagging means that we employ one more parsers during the bagging, similarly the concept can be extended to one-view bagging, two-view bagging and three-view bagging.

	weighting	UAS	LAS
one-view bagging	B	93.44	91.17
two-view bagging	B:M(5:4)	94.16	92.09
three-view bagging	B:M:J(5:3:1)	94.21	91.99

Table 4: Performance of multi-view bagging model on WSJ developing set. B denotes Berkeley parser, M denotes Mate parser and J denotes Jun parser. 5:4 and 5:3:1 denotes the best weight during voting respectively.

In the end, The three-view bagging is chosen as our final system which we use *HIT$_{baseline}$* to denote. We give the results of the system on the six domains' testing section in Table 7.

3.3 Utilization of Unlabeled Corpus

We also tried to utilize the unlabeled data to improve the final performance of our system via tri-training. Let L denote the labeled data set and U denote the unlabeled data set. Assume that three parsers P_1, P_2 and P_3 have been trained on L. Each iteration if the output results of P_2 agree on that of P_3 for one sentence, then the sentence with the output results is added to training data set of P_1. It is similar to obtain the automate training data of P_2 and P_3.

As Berkeley parser needs corpus of constituent structures to train and it is hard to transform dependency structures to constituent structures, we can't generate training corpus for it. We only did a try to generate auto-labeled training corpus for Mate parser since the performance of transition parser was not good. If the dependency structures without labels of transition parser are consistent with that of Berkeley parser, the auto-parsed result of Berkeley parser was added to training set of Mate parser. Table 5 shows the performance of one-view bagging model for Mate parser on weblogs and emails developing set. Mate Bagging$_{\text{baseline}}$ denotes the single-view bagging model of Mate parser with only gold training corpus, and Mate Bagging$_{\text{domain}}$ denotes the model with additional auto labeled corpus.

corpus	method	UAS	LAS
emails	Mate Bagging$_{\text{baseline}}$	82.00	77.75
	Mate Bagging$_{\text{domain}}$	82.43	78.42
weblogs	Mate Bagging$_{\text{baseline}}$	89.55	86.51
	Mate Bagging$_{\text{domain}}$	90.42	87.59

Table 5: Tri-training Result on emails developing set.

As is shown in Table 5, the values of LAS obtains an average improvement of 0.88%. However, the performance doesn't always achieve such a gain on the three-view bagging model which is shown in Table 6. *HIT$_{domain}$* denotes our final three-view bagging system with additional auto labeled corpus.

Finally, we give the results of our system on the six domains' testing section in Table 7. We can see that the improvements after applying tri-training are non-significant.

corpus	method	UAS	LAS
emails	HIT$_{\text{baseline}}$	83.94	79.71
	HIT$_{\text{domain}}$	83.63	79.48
weblogs	HIT$_{\text{baseline}}$	91.35	88.52
	HIT$_{\text{domain}}$	91.51	88.65

Table 6: Tri-training Result on emails and weblogs developing set.

Corpus	Method	LAS	UAS	POS
answers	HIT$_{\text{baseline}}$	80.75	85.84	90.99
	HIT$_{\text{domain}}$	80.79	85.86	90.99
emails	HIT$_{\text{baseline}}$	78.94	83.21	88.79
	HIT$_{\text{domain}}$	78.58	82.79	88.79
newsgroups	HIT$_{\text{baseline}}$	85.26	88.90	92.32
	HIT$_{\text{domain}}$	85.18	88.81	92.32
reviews	HIT$_{\text{baseline}}$	81.60	86.60	90.65
	HIT$_{\text{domain}}$	81.92	86.80	90.65
weblogs	HIT$_{\text{baseline}}$	85.91	90.27	93.32
	HIT$_{\text{domain}}$	85.89	89.43	93.32
WSJ	HIT$_{\text{baseline}}$	91.88	93.88	97.76
	HIT$_{\text{domain}}$	91.82	93.83	97.76

Table 7: Final results of our system.

4 Conclusion

We described our system of *Shared Task on Parsing the Web*. We exploited a multi-view bagging method, and obtain the best accuracy on WSJ testing corpus. Tri-training is used to generate automated training set from unlabeled corpus to improve performance of our system on other domains, and unfortunately we failed to obtain some gains.

In the future, we will analyze the reason for our failure to apply tri-training method and try better approaches for this task.

Acknowledgments

We especially thank Weiwei Sun for her suggestion of bagging for system-combination in this work. 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

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, number August, pages 89–97.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Li Ming and Zhi-Hua Zhou. 2005. Tri-training: exploiting unlabeled data using three lassifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

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.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing*.

Weiwei Sun. 2012. *Learning Chinese Language Structures with Multiple Views*. Ph.D. thesis, Saarland University.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.