Dependency Parsing for Chinese Long Sentence: A Second-stage Main Structure Parsing Method

Bo Li
School of informatics
University of Edinburgh
11 Crichton St, Edinburgh
EH8 9LE,UK
libo.whu@gmail.com

YunFei Long
School of Computer Science
Nanjing Normal University
No 1, Wen Yuan Road, Nanjing, China
893997052@qq.com

WeiGuang Qu
School of Computer Science
Nanjing Normal University
No 1, Wen Yuan Road,
Nanjing, China
wgqu_nj@163.com

Abstract

This paper explores the problem of parsing Chinese long sentences. Inspired by human sentence processing, a second-stage parsing method, referred as main structure parsing in this paper, are proposed to improve the parsing performance as well as maintaining its high accuracy and efficiency on Chinese long sentences. Three different methods have attempted in this paper and the result shows that the best performance comes from the method using Chinese comma as the boundary of the sub - sentence. According to our experiment about testing on the Chinese dependency Treebank 1.0 data, it improves long dependency accuracy by around 6.0% than the baseline parser and 3.2% than the previous best model.

1 Introduction

In recent years, the transition-based dependency parsing has been a hot research topic in Chinese parsing because of suitable to Chinese grammar profile and its linear scale time complexity. (Zhou, 2000) (Nivre and McDonald, 2008). However, although transition-based dependency parsing research has made great progress with the state-of-art performing at around 86% accuracy (Nivre et al., 2011), it still faces some problems when parsing Chinese long sentences.

First, the parser performance decreases when the length of input Chinese sentence increases. In other words, it cannot parse Chinese long sentences as accurate as short ones. As a result, if there are more long sentences in the input sentences, the overall accuracy will be affected significantly. The experiments in this paper on sentences of different length ranges show that the overall accuracy will decrease more than 1% when the length of input sentences is more than 50. This phenomenon is not only present in Chinese long sentences, but also found during parsing research of other languages such as English and French (Candito et al 2012).

The second problem is that long sentences always contain global ambiguities, and the inaccuracies on long sentences can lead to a very different understanding of a sentence. While the short sentences have more local ambiguity and inaccuracies on short sentences normally, only cause misunderstanding on details. This is because long sentences tend to contain more details about semantic and discourse information compared with short sentences. Those details confuse parsers and prevent them from finding out what the correct structure of the long sentence.

Although the reasons that should be responsible for the performance decrease in parsing long sentences are still controversial, a common explanation is that there are some rarely seen features in long sentences causes the degraded performance (Candito et al., 2012).

Unfortunately, these features cannot be learnt by transition-based parser via increasing the scale of training corpus, because the idea of the transition-based dependency parsing methods is to process a sentence incrementally, some global information from those input sentences has been neglected during the process. Attempts to include that global information in transition-based dependency parsing have been made in past years (Nivre and McDonald, 2008; Nivre et al., 2011), but those methods always have to make a tradeoff between accuracy and efficiency. What this paper tries to propose is a parsing method that achieves better performance when parsing Chinese long sentences and freezes the O (n) time complexity simultaneously.

The fact that humans can understand a long sentence correctly even when some words are unknown is quite inspiring. It implies that not all words are equally important in terms of understanding a sentence. Some words carry more syn-

29th Pacific Asia Conference on Language, Information and Computation: Posters, pages 337 - 344
Shanghai, China, October 30 - November 1, 2015

tactic and semantic information than others during people sentence understanding. Errors in recognizing those words may lead to understanding problems.

This is also true for dependency parsers. The reason why it cannot parse long sentence accurately is it does not distinguish those words from all words in a long sentence. In short sentence case, those words are always can be found because the pattern between those words is limited, which means a large training corpus can almost cover all the patterns between words, but that does not work well in long sentences. On one hand, as the input sentence gets longer, the possible combinations between words will outnumber the patterns can be found in the training corpus. On the other hand, there will be sentence-level instead of only word-level combination in a long sentence, which is beyond the transition-based parsing mode.

Therefore, this paper proposes a two-stage parsing method to help parsers find out those important words in sentences and use the information to improve parsing performance with out at the expense of time complexity.

2 Related Work

Dominating dependency-parsing models can be categorized into three families: graph-based models (Eisner, 1996; McDonald et al., 2005; Mc-Donald and Pereira, 2006; Wang et al., 2007; Zhang and Clark, 2008), transition-based models (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004) and hybrid models (Sagae and Lavie, 2006; Nivre and McDonald, 2008; Zhang and Clark, 2008).

The advantage of the graph-based parsing is that it processes the input sentence as a whole. In other words, it takes global information of the input sentence into consideration, which gives it a higher accuracy on average than other models (Nivre, 2007) However, because of adopting global information, the efficiency of graph-based parsing models are comparatively lower (O(n^2)) as the searching space is much larger.

By contrast, transition-based model, which is also referred as action-based parsing model, significantly outperform in efficiency. The transition-based parsing is essentially a discriminative algorithm which processes words incrementally. According to (Nivre and McDonald, 2008), transition-based parsing gives time complexity as low as O (n) (projective situation).

In Chinese dependency parsing research, transition-based parsing is a preferable choice because it suits better with the syntax of Chinese (Lai and Huang, 1994; Lai et al., 2001; Wang, William Yang, et al., 2014). Compared with English dependency parsing, Chinese dependency parsing is slightly underperformed. That is partially because there are a few widely used Chinese dependency corpus. The Penn Chinese TreeBank (CTB) is a promising choice. However, it is still not complete enough compared with that in English. For performance evaluation, Nivre (2011) provide a widely accepted comparison result, according to this paper, the state-of-art performance of Chinese dependency parsing is around 86.0% in unlabeled attachment scores (UAS).

Some recent research on improving the Chinese parsing performance by introducing multiple layer parsing approach (Ping Jian, et al., 2009) has been made, but it does not consider Chinese features. Zhenghua et al (2010) proposed the idea of using punctuation to help improving parsing, which also been discussed in this paper. However, the major difference is that punctuation is just one perspective of the framework proposed in this paper. In addition, this paper achieves a better performance compared with previous works.

3 A New Framework

3.1 Framework Design and Parsing Process

As previous discussion, the key to parse long sentences accurately is to find out the words that carry structural information about the sentence, which named as the main structure words in this paper. However, the challenge is that normal transition-based parsing methods cannot find out those main structure words because of lack of global information. In this circumstance, we select the output of a transition-based parsing method, which contains candidate features for main structure word recognition, after main structure word recognition, a second transition-based parser, which trained in a special corpus, introduced to adjust the dependencies between those main structure words. This second-stage parsing method referred as main structure parsing.

The purpose of the first parsing stage is to find out main structure words. In the first parsing stage, the baseline parser parses the input sentence in the normal way. From the output of baseline parser, the information for finding out main structure words extracted by following cer-

tain steps in the framework, and then the information is pass into the next parsing stage.

The information obtained from baseline parser is:

Short Dependencies: The short dependencies are normally from words that occur in the same sub-sentence (sub-sentence is the part between two punctuations in a sentence; a long sentence normally consists of multiple sub-sentences). The structure of these dependencies tends to be less complicated, and traditional transition-based parser can achieve over 90% accuracy on these dependencies. As the accuracy for these short dependencies is high, they are assumed as correct dependencies within the sub-sentence. Therefore, in the next stage, the main structure parser can only focus on these long dependencies, which is the key idea of the framework.

Long Dependencies: The long dependencies normally occur between words from different sub-sentences. The words that carry long dependencies are potential main structure words; normally they determine the global structure of the whole sentence. However, as the dependencies between main structure words are much longer than normal dependencies, traditional transition-based parsers are inaccurate on them. Given this, this paper uses a specially trained parser to re-parse the long dependencies regardless of the short dependencies. The result can be merged with short dependencies from first stage parsing through a voting scheme.

Other Information: Including the length of the input sentence, the number of sub-sentences, etc.

The challenge after obtaining the three kinds of information is how to distinguish actual main structure words from these potential ones. Three different methods are proposed and discussed in the paper in Chapter 3.2.

The goal of the second parsing stage is to find out correct dependencies between the main structure words. From previous discussion, the reason why a transition-based parser cannot parse main structure words correctly in long sentence is that syntactically redundant detail brings significant ambiguity. Therefore, in this stage, those details are ignored temporarily and only main structure words are processed. Obviously, a parser trained in normal corpus is not able to parse main structure words directly. The parser used in the second stage will be trained in a special corpus that only contains main structure words. As there is no available corpus like this, this paper adopts a special training corpus produced by the automatic main structure words extraction method introduced in Chapter 3.2.

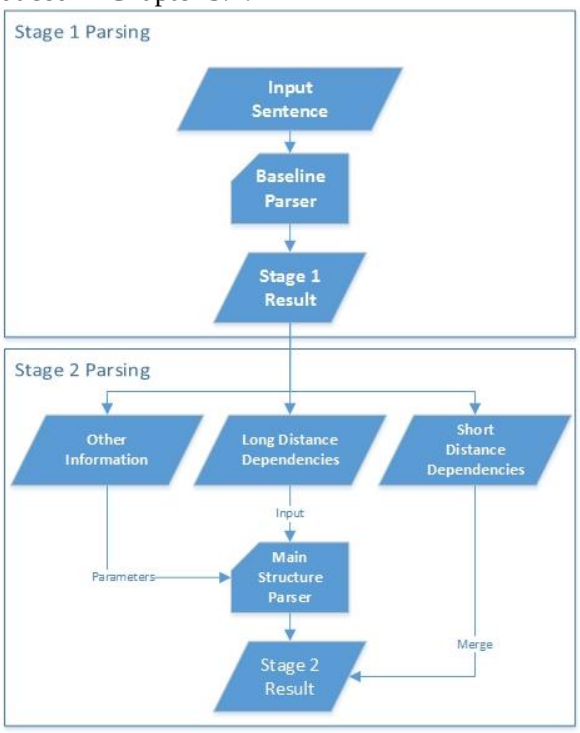


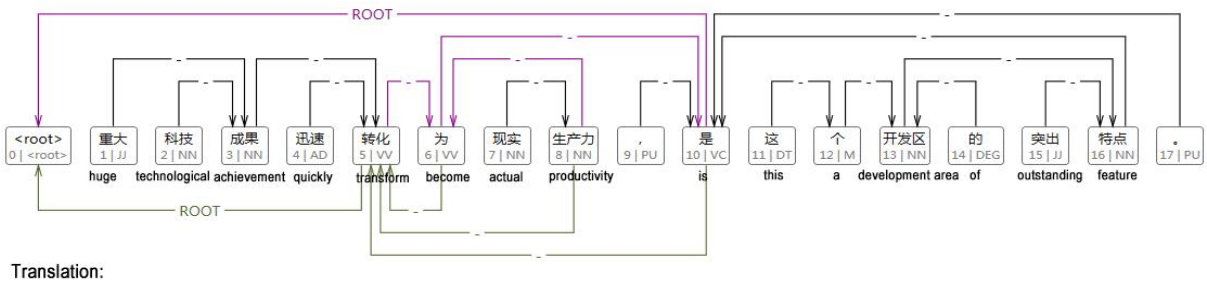
Figure 1: Framework of parser

3.2 Automatic Main Structure Words Extraction

Although main structure words are necessary for sentences, it is not easy to extract those words automatically. The differences between main structure words and non-main-structure words provide features to distinguish them.

Since it takes at least two components, namely head and its dependency, to form a dependency unit in a sentence, the parsing method also tries to find features of the main structure words from the two perspectives.

From the head perspective, the main structure words are normally the center constituent of its sub-sentence. The dependency relation between words could be regarded as a voting action. The more votes a word receives from its neighbor words, the more important the word is. Given the main structure words are usually the most important (important to the sentence structure) words, they tend to receive more votes from other words in its sub-sentence. Figure 2 (a) shows the process, the word 'Chengwei'(Becoming) and 'Touzi'(Investment) which have more incoming dependencies, are selected as main structure words from an example sentence (Figure 2 (a)).



Translation:

The fact that huge technological achievements can be quiculy transformed into actual productivity, is the outstanding feature of this development area.

Figure 2 (a): An example sentence

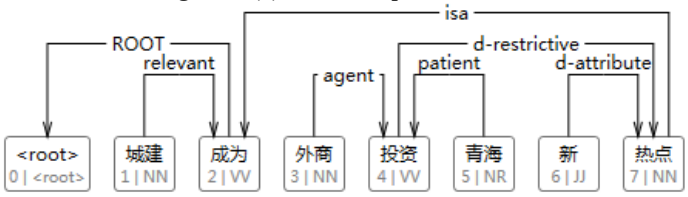


Figure 2 (b): word 'ChengWei' and 'TouZi' have more votes

From the dependency perspective, the main structure words are those words with long distance dependency. As main structure words are distributed among sub-sentences, the dependencies between main structure words are supposed to cross sub-sentences instead of being within a sub-sentence.

As previous discussion, three selection criteria that are significant in selecting main structure words are proposed: Crossing comma, the number of incoming dependencies and length of dependency. Based on the selection criteria, the following three methods for automatic main structure selection are attempted.

Method 1: Crossing comma

The idea of this method is to check whether a comma lines between the dependency head and its dependency. If the comma exists, both head and its dependency have been identified as main structure words.

The reason is that Chinese comma tends to represent weak stop between sub-sentences compared with that the English comma tends to represent weak stop between words or terms. In other words, the sub-sentences separated by commas are potential independent sentences. They just happen to be connected by commas because of the expression convention in Chinese. Given this, dependency that crosses a comma is of high chance to be between words that control structure of the sentence. Otherwise, the two sub-sentences are disconnected syntactically. This situation is very common, especially in Chinese long sentences.

The process of the extraction is as follows.

Step 1: For each word w_i In a sentence S $(w_0, w_1, w_2 \ldots w_n)$, find out all the incoming dependencies D_{ji} (dependency starting at word W_j and ending at word W_i, i ≠ j).

Step 2: For each incoming dependency D_{ji}, check if there is a word w_k(k ≠ i, j) equal to the character comma (",").

Step 3: If there such a word w_k, the word w_i is selected as main structure word.

Figure 3 shows the result of the selecting method on the example sentence (Figure 2 (a)).

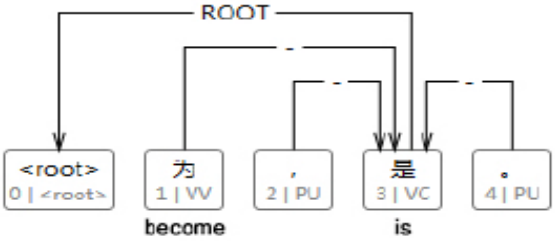


Figure 3: An example of Method 1

Method 2: the number of incoming dependencies

The idea behind this method is that the main structure words are always the center (root) of each sub-sentences, most other non-main structure words act as their modifier. Therefore, main structure words tend to have the most income dependencies from other words in each sub-sentence. The main structure words can be found

by counting the number of dependencies coming from other words.

Given that the number of sentences varies from sentence to sentence, the percentage instead of numbers are used as the measurement. The percentage is calculated within each sub-sentence rather than the whole sentence because the income dependency method does not work for long distance dependency. The process is as follows.

Step 1: For a sentence S $(w_0, w_1, w_2 \dots w_n)$, split it into sub-sentences by the character comma (","). The sub-sentences are:

$$S_{sub1}(w_0, w_1, w_2 \dots w_{i-1}),$$
$$S_{sub2}(w_{i+1}, w_{i+2}, w_{i+3} \dots w_{j-1})$$
$$\dots$$
$$S_{subn}(w_{n-k+1}, w_{n-k+2}, w_{n-k+3} \dots w_n)$$

Step 2: For each sub-sentence S_{subi}, calculate the number of words within the sub-sentence N_{subi}.

Step 3: For each word w_j in the sub-sentence S_{subi}, calculate the number of incoming dependencies N_{inj}.

Step 4: For each word w_j in the sub-sentence S_{subi}, calculate the percentage $\mathrm{p}(w_j) = \frac{N_{inj}}{N_{subi}}$.

Step 5: Compare $\mathrm{p}(w_j)$ with pre-fixed threshold p, if $\mathrm{p} \leq \mathrm{p}(w_j)$, the word w_j is selected as main structure word.

Method 3: Length of dependency

This method uses the length of dependency as threshold to identify main structure words; this is enlightened by our observation that normally non-main structure words have short distance dependency because their dependencies are within the sub-sentence. Main structure words have dependencies with much longer lengths, so those words with length longer than normal situation are regarded as main structure words. The process is as follows.

Step 1: For each word w_i In a sentence S $(w_0, w_1, w_2 \dots w_{n-1})$, find out all the incoming dependencies D_{ji} (dependency starting at word W_j and ending at word W_i, i ≠ j).

Step 2: calculate the distance between i and j $Dis_{ij} = |i - j|$.

Step 3: Normalize the Distance Dis_{ij} by diving the whole length of sentence S, get length percentage$\mathrm{P}(w_i) = (Dis_{ij}/n) * 100\%$.

Step 4: Compare $\mathrm{p}(w_i)$ with pre-fixed threshold p, if$\mathrm{p} \leq \mathrm{p}(w_j)$, the word w_j is selected as main structure word.

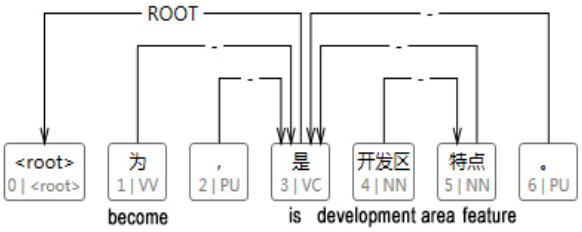


Figure 4: An example of Method 3

As there is a threshold parameter significantly affecting the performance in method 2 and method 3, a range of parameters examined over the whole testing set to find out the one with the best performance. Then our experiment was run over different length testing set to explore the best improvement it brings to baseline performance, which will, demonstrated in chapter 4.

3.3 Dependency Voting

According to previous discussion, the first stage parsing achieves high accuracy (around 90%) on short dependencies and low accuracy (around 30%) on long dependencies, while the second stage parsing has significantly better performance (around 40%, will discuss it in an experiment) on long distance dependencies. Some words (mostly main structure words) may have two parsing results, one from first stage parsing, and the other from second stage parsing. This paper uses a weighted voting scheme to decide what the final dependency for the words is. The weight of each parser comes from its accuracy on those specific parts. For example, baseline parser achieves around 84% accuracy in short distance dependency; when it predicates a short distance dependency, there are 84% possibility that the dependency is right. Each word receives two predicates from the two parsers; if the two predicates are the same, the result is the dependency. Otherwise, the parser with higher accuracy wins. That means the dependency is determined by the result from a more reliable parser. In this case, the two parser voting can be simplified as that baseline parser controls short distance dependencies parsing result, while the second stage parser controls the result of long distance dependency parsing.

4 Result Analysis

4.1 Corpus

We train and evaluate our parser on the dependency corpus called Chinese dependency Treebank 1.0 from Harbin Institute of Technology

(HIT). This corpus is available on the webpage of the conference of natural language processing and Chinese computing (NLPCC). Corpus follows the CoNLL2007 Standard, contains about 8,301 sentences in the training set, 534 sentences in the development set and 1,233 sentences in the test set. It has a much longer average sentence length than Penn Chinese Treebank (PCTB) (33 compare to 28 (Xue et al., 2005). For all experiments, we use the test set and report unlabeled attachment scores (UAS) for evaluation.

4.2 Baseline Parser

The baseline parser used in this paper is the Malt parser proposed by (Nivre, 2007). Based on the previous analysis, the state-of-art graph-based parser is slightly outperformed than transition-based parser while at the expense of surging scale of efficiency to O (n^2). We aim to improve transition-based parser in order to maintain O (n) efficiency while improve accuracy as much as possible. In other words, there are other advanced parsers giving better performance than Malt parser in terms of accuracy, they are not selected because the processing speed in these parsers is sacrificed more or less. Compared with short sentence parsing, the importance of parsing efficiency (processing speed) in long sentence parsing is more significant. From this perspective, the Malt parser, providing O (n) efficiency with a little cost of accuracy is an ideal choice in this paper.

4.3 Experiments

Table 1 shows the test results of our parser on short dependencies. We include in the table results from the pure transition-based parser of (Zhang and Clark, 2008), the dynamic-programming arc-standard parser of (Huang and Sagae, 2010) and parsing with rich non-local features of (Zhang and Nirve, 2011) on Chinese. Our baseline parser and its extended methods are very close to its competing parsers in terms of the performance on short dependencies.

Table 2 shows the results of our parser on long dependencies (The dependencies between main structure words). Normally, the main structure words located in different sub-sentences of a long sentence. As a result, normal transition-based parsers cannot handle them well, which is the reason for the low scores overall. Our scores for this test set are the best reported so far and significantly better than the previous systems. In all our three methods, the best result is from the method 1, which improves the performance on long dependencies by around 6.2% from baseline parser, while outperformance previous best system by 3.2%.

	UAS
Baseline	84.6%
Baseline +Method 1	84.3%
Baseline +Method 2	84.2%
Baseline +Method 3	84.4%
Zhang and Clark 2008	84.3%
Huang and Sagae 2010	85.2%
Zhang, Y., & Nivre 2011	86.0%

Table 1: Performance on short dependencies

	UAS
Baseline	32.1%
Baseline +Method 1	38.3%
Baseline +Method 2	36.6%
Baseline +Method 3	37.0%
Zhang and Clark 2008	33.3%
Huang and Sagae 2010	32.9%
Zhang, Y, & Nivre 2011	35.1%

Table 2: Performance on long dependencies

4.4 Parameter Optimization

In the three methods, the performance of method 2 and method 3 largely affected by threshold parameters, Table 3 shows the relationship between the threshold and accuracy, the best performance of method 2 achieved when the threshold set to be 35%.

Threshold	Accuracy	Threshold	Accuracy
1.00	29.5%	0.50	34.0%
0.95	29.6%	0.45	34.3%
0.90	29.8%	0.40	35.4%
0.85	29.8%	**0.35**	**36.1%**
0.80	30.0%	0.30	35.3%
0.75	30.3%	0.25	33.3%
0.70	30.9%	0.20	31.8%
0.65	30.7%	0.15	30.7%
0.60	32.0%	0.10	28.2%
0.55	33.1%	0.05	26.4%

Table 3: Performance of Method 2 with Different Threshold

The method 2 achieves the best performance when there are not enough words chosen as main structure words. For example, the 1.00 means that one word has to get all dependencies from other words, within its sub-sentence to be selected as the main structure word, and this is almost impossible. Lower the threshold also deteriorates

the performance because that means more words are chosen as main structure words. For example, the 0.00 means all words are main structure words, and they are all parsed by stage 2 parsers which training in a main structure corpus.

Threshold	Accuracy	Threshold	Accuracy
0	11.7%	26	33.0%
2	21.1%	28	32.6%
4	27.1%	30	32.6%
6	30.9%	32	32.4%
8	32.9%	34	32.3%
10	32.6%	36	32.3%
12	33.5%	38	32.2%
14	33.6%	40	32.2%
16	**33.8%**	42	32.1%
18	33.4%	44	32.1%
20	33.3%	46	32.0%
24	33.3%	48	32.0%

Table 4: Performance of Method 3 with Different Threshold

Table 4 shows the performances of the main structure parser with method 3 with different parameters. The method achieves the best performance when the parameter is set to 16, the performance curve before and after this point experience a comparable decrease like method 2.

Length	Baseline	M1	M2	M3
40-50	33.2%	**40.0%**	38.2%	33.4%
50-60	32.4%	36.4%	35.5%	32.0%
60-70	32.2%	36.9%	**37.2%**	33.2%
70-80	28.8%	34.5%	35.2%	31.8%
80-90	22.0%	30.2%	29.1%	26.8%
90-100	23.5%	30.5%	26.5%	28.1%
100-110	26.1%	29.1%	26.7%	**32.0%**
110-120	25.7%	29.2%	28.4%	31.1%
120-130	19.0%	22.5%	21.7%	24.4%
130-140	21.1%	32.1%	22.2%	27.8%
140-150	20.0%	21.9%	27.1%	24.3%

Table 5: Method comparison on each sub-set

As can be seen from section 4.4, method 1 outperformed the other two methods in both UAS and ULAS, Table 5 shows that the method 3 achieves better performance than method 1 when the length of sentence is longer than 100, while it is worse than method 2 in sentences with length less than 80.

5 Conclusion

This research proposes a two-stage parsing method called main structure parsing, used to improve the parsing performance for Chinese long sentence. The performance of normal dependency parser decreases on long sentences because of the long distance dependencies between main structure words. The main structure parsing method alleviates long dependency problem by selecting out the main structure words and parse them with a specially trained parser. Three different methods regarding selecting those main structure words are compared and tested in the thesis. The best method achieves a 6.0 % improvement on long dependency than baseline parser and 3.2% improvement than the previous best mode.

Reference

Jones, B. (1996). What's the point? A (computational) theory of punctuation.

Bohnet, B., & Kuhn, J. (2012, April). The best of both worlds: a graph-based completion model for transition-based parsers. In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (pp. 77-87). Association for Computational Linguistics.

Candito, M., & Seddah, D. (2012, November). Effectively long-distance dependencies in French: annotation and parsing evaluation. In TLT 11-The 11th International Workshop on Treebanks and Linguistic Theories.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology, 2: 27: 1–27: 27, 2011. Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm.

Zhao, Y., Che, W., Guo, H., Qin, B., Su, Z., & Liu, T. (2014). Sentence compression for target-polarity word collocation extraction. In Proceedings of COLING (pp. 1360-1369).

Che, W., Spitkovsky, V. I., & Liu, T. (2012, July). A comparison of chinese parsers for stanford dependencies. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2 (pp. 11-16). Association for Computational Linguistics.

Ferreira, F., Engelhardt, P. E., & Jones, M. W. (2009). Good enough language processing: A satisficing approach. In Proceedings of the 31st Annual conference of the Cognitive Science Society. Austin: Cognitive Science Society.

Yamada, H., & Matsumoto, Y. (2003, April). Statistical dependency analysis with support vector machines. In Proceedings of IWPT (Vol. 3, pp. 195-206).

Lai, B. Y. T., & Huang, C. (1994). Dependency grammar and the parsing of Chinese sentences. arXiv preprint cmp-lg/9412001.Lai, T. B., Huang, C., Zhou, M., Miao, J., Siu, T. K., 2001. Span-based statistical dependency parsing of Chinese. In: NLPRS. pp. 677–684.

Li, X., Zong, C., & Hu, R. (2005). A Hierarchical Parsing Approach with Punctuation Processing for Long Sentence Sentences. In In Proceedings of the Second International Joint Conference on Natural Language Processing: Companion Volume including Posters/Demos and Tutorial Abstracts.

Li, Z., Che, W., & Liu, T. (2010, December). Improving dependency parsing using punctuation. In Asian Language Processing (IALP), 2010 International Conference on (pp. 53-56). IEEE.

Xun Jin, M., Kim, M. Y., Kim, D., & Lee, J. H. (2004). Segmentation of Chinese long sentences using commas. In Proceedings of SIGHAN (pp. 1-8).

Covington, M. A. (2001). A fundamental algorithm for dependency parsing. InProceedings of the 39th annual ACM southeast conference (pp. 95-102).

Nivre, J., & McDonald, R. T. (2008, June). Integrating Graph-Based and Transition-Based Dependency Parsers. In ACL (pp. 950-958).

Nivre, J., Hall, J., & Nilsson, J. (2006, May). Maltparser: A data-driven parser-generator for dependency parsing. In Proceedings of LREC (Vol. 6, pp. 2216-2219).

Nilsson, J., Riedel, S., & Yuret, D. (2007, June). The CoNLL 2007 shared task on dependency parsing. In Proceedings of the CoNLL shared task session of EMNLP-CoNLL (pp. 915-932).

Nivre, J., & McDonald, R. T. (2008, June). Integrating Graph-Based and Transition-Based Dependency Parsers. In ACL (pp. 950-958).

MAO, Q., LIAN, L. X., ZHOU, W. C., & YUAN, C. F. (2007). Chinese syntactic parsing algorithm based on segmentation of punctuation. Journal of Chinese Information Processing, 21(2), 3.

Sagae,K and Lavie,A. 2006a. Parser combination by reparsing. In Proc. HLT/NAACL, pages 129–132, New York City, USA, June.

Sagae, K., & Lavie, A. (2006, June). Parser combination by reparsing. InProceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers (pp. 129-132). Association for Computational Linguistics.

Wang, W. Y., Kong, L., Mazaitis, K., & Cohen, W. W. (2014). Dependency Parsing for Weibo: An Efficient Probabilistic Logic Programming Approach. Association for Computational Linguistics.

Xue, N., Xia, F., Chiou, F. D., & Palmer, M. (2005). The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. Natural language engineering, 11(02), 207-238.

Zhou, M. (2000, October). A block-based robust dependency parser for unrestricted Chinese text. In Proceedings of the second workshop on Chinese language processing: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 12 (pp. 78-84). Association for Computational Linguistics.

Zhang, Y., & Nivre, J. (2011, June). 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: short papers-Volume 2 (pp. 188-193). Association for Computational Linguistics.