Improving Dependency Parsing Using Punctuation

Zhenghua Li, Wanxiang Che, Ting Liu
Research Center for Information Retrieval, School of Computer Science and Technology
Harbin Institute of Technology, Harbin, China
{*zhli,car,tliu*}*@ir.hit.edu.cn*

***Abstract*—The high-order graph-based dependency parsing model achieves state-of-the-art accuracy by incorporating rich feature representations. However, its parsing efficiency and accuracy degrades dramatically when the input sentence gets longer. This paper presents a novel two-stage method to improve high-order graph-based parsing, which uses punctuation, such as commas and semicolons, to segment the input sentence into fragments, and then applies a two-level parsing. Experimental results on the Chinese data set of the CoNLL 2009 shared task [1] show that our two-stage method significantly outperforms both the conventional one-stage method and previously-proposed three-stage method in terms of both parsing efficiency and accuracy.**

***Keywords*-dependency parsing; graph-based; punctuation;**

I. Introduction

Given an input sentence $x = w_0, w_1, ..., w_n$, the goal of dependency parsing is to construct a labeled dependency tree of the kind depicted in Figure 1. A labeled arc is represented as a triple (h, m, l), meaning node h is the *head* (or *father*), node m is the *modifier* (or *child*, *dependent*), and their syntactic relation is l. A dependency tree must satisfy the constraint: each word must have exactly one incoming arc. Node 0 is a pseudo-node which points to the *root* of the sentence.

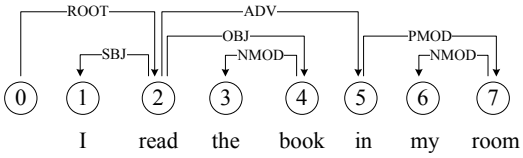


Figure 1. An example dependency tree.

Graph-based parsing views the problem as finding the highest scoring tree $y^*(G_x)$ from a directed multigraph $G_x = (V_x, A_x)$, where the node set V_x represents all the words in x, and the arc set A_x represents all the arcs in the form of (h, m, l). Carreras proposed a high-order graph-based model[2]. In his model, the score of a dependency tree is decomposed as the following formula.

$$s(y) = \sum s_{arc}(h, m, l) + \sum s_{sib}(h, m', m, l) + \sum s_{grd}(h, m, g, l)$$

Where $s_{arc}(h, m, l)$ represents the score of a single arc (h, m, l) in the tree; $s_{sib}(h, m', m, l)$ is the score of a sibling arc pair, i.e. $(2, 4, \text{OBJ})$ and $(2, 5, \text{ADV})$ in Figure 1; $s_{grd}(h, m, g, l)$ is the score of a grand-parental arc pair, i.e. $(2, 5, \text{ADV})$ and $(5, 7, \text{PMOD})$ in Figure 1. The high-order model can significantly improve parsing accuracy, but its time complexity is $O(n^4)$.

To alleviate the efficiency problem of high-order graph-based parsing, we propose a fragment-based two-stage approach. Punctuation is used to segment a sentence into fragments. In the first stage, all fragments are independently parsed to obtain their structures. In a fragment, there may exist multiple words which link with words outside the fragment. To represent these words, we allow the structure of a fragment to have multiple roots. In the second stage, the roots of all fragments along with punctuation are parsed to obtain the inter-fragment structure. Experimental results show that our two-stage method improves the high-order graph-based parsing in terms of both parsing efficiency and accuracy.

The central idea of this paper is to segment a sentence into fragments with punctuation and then apply a two-level parsing. The main challenge for this idea is that parsing accuracy suffers when the segmentation produces fragments that do not correspond to a complete syntactic tree. Two kinds of previous work have been done to overcome this problem, but they all suffer from some weakness. (1) The methods of [3], [4] used classifiers to recognize the fragments which corresponds to complete syntactic trees. However, the classifiers perform badly due to the hardness of this problem. We will introduce the method of [4] in detail and compare it with our method later. (2) The method of [5] based on some heuristic rules, which can only be applied to constituent parsing.

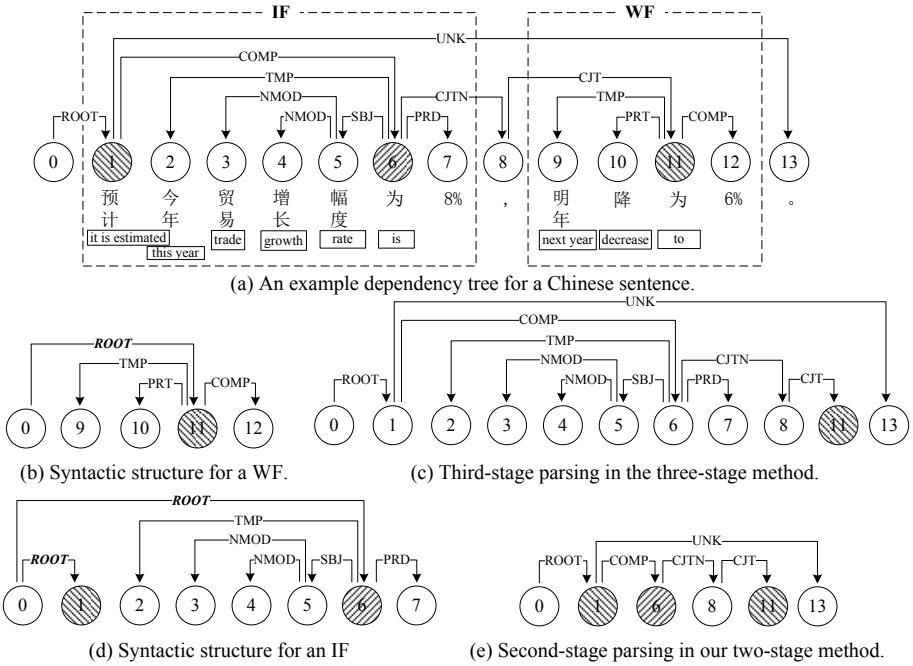
The remainder of this paper is organized as follows. Section II describes approaches based on sentence fragmentation in detail. Section III presents the experiments and discussions. Section IV concludes this paper.

II. Approaches Based on Sentence Fragmentation

A. Formal Definitions

The following are some terminology that we will be using.

Split punctuation: Based on previous work [3], [5], [4] and our statistical analysis on the Chinese treebank, the following punctuation, including commas, colons, semicolons, periods, question marks and exclamation marks, are used as sentence fragmentation points. We call them *split punctuation*. According to these split punctuation, nearly 90% of the resulting fragments correspond to complete syntactic trees (well-formed). Other punctuation



(a) An example dependency tree for a Chinese sentence.

(b) Syntactic structure for a WF.

(c) Third-stage parsing in the three-stage method.

(d) Syntactic structure for an IF

(e) Second-stage parsing in our two-stage method.

Figure 2. An example sentence and its dependency tree taken from the development set. The translation is "It is estimated that the trade growth rate is 8% this year, and will decrease to 6% next year." The shaded nodes indicate sub-roots.

marks, such as quotation and pause marks, are not used, as they do not have this characteristic.

Fragment: A sentence is segmented into several parts according to split punctuation. Each part is called a fragment. As shown in Figure 2-(a), two fragments, i.e. nodes 1-7 and nodes 9-12, are produced.

Sub-root: If a node in a fragment depends on a node (father) outside the fragment, or has a *descendant* (child, grandchild, or further) outside the fragment, it is called a *sub-root* of the fragment. The former case is called *type 1*, such as node 1 and 11 in Figure 2-(a). The latter case is called *type 2*, such as node 6.

Well-formed Fragment (abbreviated as *WF* later): A fragment which contains only one sub-root, e.g., the second fragment in Figure 2-(a), is called a WF. The syntactic structure of a WF is complete, meaning that it contains all modifiers and heads of all words in the fragment, except for the sub-root.

Ill-formed Fragment (*IF*): A fragment which contains more than one sub-roots, e.g., the first fragment in Figure 2-(a), is called an IF.

B. The Three-stage Approach

The method of [4] was originally designed for constituent parsing. We make minor modifications when applying it to dependency parsing.

Stage 1: IF-WF classification Each fragment in the sentence is classified as an IF or a WF.

Stage 2: parsing WFs All WFs in the sentence are independently parsed. Figure 2-(b) shows the resulting structure of the WF in Figure 2-(a). According to our definition of WFs, the syntactic structure has one root node, which corresponds to the sub-root of the WF.

Stage 3: parsing the sentence Since all WFs' structures are obtained, we only need to parse the pseudo-sentence, which contains IFs' nodes, WFs' sub-roots, and split punctuation, as shown in Figure 2-(c).

C. Our Two-stage Approach

Our two-stage method avoids the IF-WF classification of the three-stage method, and tries to find those words that link with words outside the fragment during the first-stage parsing.

Stage 1: parsing fragments All fragments in the input sentence are independently parsed. The resulting structures are shown in Figure 2-(b) and (d). The syntactic structure of a fragment may contain multiple root nodes. The root nodes represent the sub-roots of the fragment.

Stage 2: parsing the sentence Since the structures of all fragments are obtained, we only need to parse the sub-roots of all fragments and split-punctuation, as shown in Figure 2-(e).

III. Experiments and Discussions

We use the Chinese data set in the CoNLL 2009 shared task [1]. The main reason why we choose Chinese treebank is that the average use of commas per sentence is more frequent in Chinese than other languages such as English [3]. We use two standard evaluation metrics in dependency parsing community. LAS, short for labeled attachment score, means the percentage of words which are given the correct head with the correct label. UAS, short for unlabeled attachment score, means the percentage of words which are given the correct head without considering the label. Dan Bikel's randomized parsing evaluation

Table I
STATISTICS OF DIFFERENT STRUCTURES AFTER SENTENCE FRAGMENTATION WITH SPLIT PUNCTUATION.

Data set	Sent	Frag	WF	Sub-root	Sub-root#2
Train	22,277	66,111	58,172	76,989	3,512
Dev	1,762	5,204	4,556	6,105	317
Test	2,556	7,724	6,844	8,912	407

Table II
PERFORMANCE OF THE MAXIMUM ENTROPY-BASED IF-WF CLASSIFIER ON THE TEST SET.

IF		WF		Total
Recall	Precision	Recall	Precision	Accuracy
46.4%	28.1%	84.8%	92.5%	80.4%

Table III
PARSING ACCURACY OF THE THREE-STAGE METHOD WITH DIFFERENT SETTING ON THE TEST SET.

IF-WF classification setting	LAS(%)	UAS(%)
WF-gold	76.45	81.62
WF-classifier	75.87	80.97
WF-all	75.61	80.48

Table IV
PERFORMANCE OF THE FIRST-STAGE PARSING WITH RESPECT TO SUB-ROOTS.

	Gold	System	Correct	Recall	Precision
Sub-root	8,912	8,527	7,251	81.4%	85.0%
Sub-root#1	8,505	-	7,144	84.0%	-
Sub-root#2	407	-	107	26.3%	-

comparator[1] is used to do significant test.

A. Sentences Fragmentation with Split Punctuation

Table I shows the number of different structures after segmenting sentences into fragments with split punctuation. Each sentence produces about 3.0 fragments on average. Nearly 90% of the fragments are WFs. About 4.6% of the sub-roots belongs to type 2 (sub-root#2).

B. The Three-stage Approach

A maximum entropy-based classifier is trained for the IF-WF classification problem [6]. The input to the classifier is a sentence with PoS tags and a fragment's index range. The classifier needs to judge whether the fragment is a WF or an IF. Feature selection and parameter setting are made on the development set. Table II shows the performance of the classifier. We see that the IFs' recall is only 46.4%, which means nearly half of IFs are misclassified as WFs. This will certainly hurt parsing accuracy of subsequent stages. Meanwhile, the IFs' precision is only 28.1%, so nearly 70% of the IFs recognized by the classifier are actually WFs. This will affect parsing efficiency of subsequent stages.

In the second stage, we need to parse all WFs recognized in the first stage, and get their syntactic structures like Figure 2-(b). To do so, we extract all WFs' partial syntactic structures from the training set. Then, we convert the structures into the form like Figure 2-(b) by letting the sub-root depend on the pseudo-node 0 with label "ROOT". Finally, we train a parser on these converted structures, which is called *WF-parser*. WF-parser is used to parse all WFs recognized in the first stage. The root node of the resulting structure are treated as the sub-root.

In the third stage, we need to parse the pseudo-sentence comprising the nodes of IFs, sub-roots of WFs and split punctuation. We train a parser on the structures of the original training set like Figure 2-(a), which is called *ALL-parser*.

[1]http://www.cis.upenn.edu/˜bikel/software.html

The results of the second and third stages are combined to form the complete structure of the input sentence. Table III shows the evaluation results on the test set. To establish the effect of errors arising during IF-WF classification, we use three test settings: all the WFs are correctly recognized (*WF-gold*); all the WFs are predicted by the classifier in the first stage (*WF-classifier*); all the fragments are treated as WFs (*WF-all*). WF-gold shows the upper bound on parsing accuracy of the three-stage method. WF-all shows the lower bound. We can see that the IF-WF classification significantly decreases parsing accuracy by 0.58% on LAS and 0.65% on UAS.

C. Our Two-stage Approach

In the first stage, we need to parse all the fragments in the input sentence, and get their syntactic structures like Figure 2-(b) and (d). To do so, we extract all fragments' partial structures from the training set. Then we convert these structures into those in Figure 2-(b) and (d) by letting the sub-roots of a fragment depend on the pseudo-node 0 with label "ROOT". Finally, we train a parser on these converted structures, which is called *FRG-parser*. FRG-parser is used to parse all the fragments in the sentence. The root nodes of the resulting structure are treated as the sub-roots of the corresponding fragment.

As the sub-roots produced in this stage will be the input for the second-stage parsing, the recall and precision of the sub-roots are presented in Table IV. In the "sub-root#1" and "sub-root#2" rows, we evaluate the recall of sub-roots of type 1 and type 2. We can see that FRG-parser performs badly on sub-roots of type 2, missing 300 of 407. Sub-roots of type 2 are difficult to find for FRG-parser. The first reason is that the training structures only contain a small number of them. The second reason is sub-roots of type 2 actually depend on nodes in the same fragment, so it is difficult for FRG-parser to prefer letting them be root nodes. Fortunately, they are in the tiny minority, so parsing accuracy of our two-stage method is only slightly affected.

Table V
ACCURACY COMPARISON ON THE TEST SET.

Methods	LAS(%)	UAS(%)
One-stage	75.76	80.55
Two-stage	**76.13**	**81.20**
Three-stage	75.87	80.97
[7]	**76.51**	-
[8]	76.11	-
[9]	75.49	-

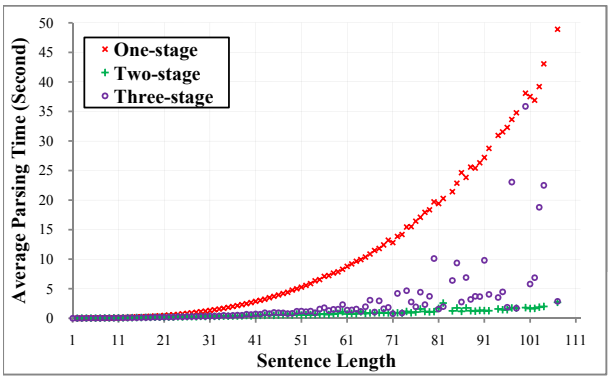


Figure 3. Efficiency comparison on the test set.

In the second stage, we need to parse the pseudo-sentence comprising sub-roots of all fragments and split punctuation. To do so, we extract structures like Figure 2-(e) from the training set, which only comprise the correct sub-roots and split punctuation in a sentence. We train a parser on these extracted structures. The goal of this parser looks like parsing the inter-fragment structure, so we call it *INTER-parser*.

The results of the above two stages are combined to construct the complete structures. Table V compares parsing accuracy of different methods. our two-stage method outperforms the one-stage method by 0.37% on LAS (p=0.02) and 0.65% on UAS (p=10^{-4}). The improvement is modest but significant. The three-stage method makes a smaller improvement of 0.11% on LAS (p=0.23) and 0.42% on UAS (p=0.003). Table V also lists accuracy of the three best parsers in the CoNLL 2009 shared task. Compared with the one-stage method, the gap from the best result is reduced by our two-stage approach.

Figure 3 compares parsing efficiency of different methods. We omit sentences longer than 110 for clarity. For the three-stage method, the time consumed by the IF-WF classifier, which is almost constant, is not counted in the figure. Besides, parsing time in the figure also includes feature construction phase. We can see that our two-stage method significantly improves the efficiency of parsing long sentences. This can lead to improved performance on high-level applications such as MT and IR, which are highly sensitive to parsing time. The three-stage method is not stable, which is affected by the errors arising from the IF-WF classification. More fragments in a sentence classified as IFs, the three-stage method will be slower.

Additionally, our two-stage approach can greatly reduce the training time by nearly 3/4, as the structures, on which FRG-parser and INTER-parser are trained, comprise much fewer nodes than those for ALL-parser.

IV. CONCLUSIONS

This paper proposes a novel two-stage approach based on sentence fragmentation for high-order graph-based dependency parsing. It has been shown to have the following benefits over both the conventional one-stage method and previously-proposed three-stage method. (1) Parsing efficiency for long sentences is significantly improved. (2) Parsing accuracy for long sentences is significantly improved. (3) The time during the training phase is also greatly reduced.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (NSFC) via grant 60803093, 60975055, the “863” National High-Tech Research and Development of China via grant 2008AA01Z144, and Natural Scientific Research Innovation Foundation in Harbin Institute of Technology (HIT.NSRIF.2009069).

REFERENCES

[1] J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang, “The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages,” in *Proceedings of CoNLL 2009*, 2009.

[2] X. Carreras, “Experiments with a higher-order projective dependency parser,” in *Proceedings of EMNLP/CoNLL*, 2007, pp. 141–150.

[3] M. Jin, M.-Y. Kim, D. Kim, and J.-H. Lee, “Segmentation of Chinese long sentences using commas,” in *ACL SIGHAN Workshop 2004*, 2004.

[4] Q. Mao, L. Lian, W. Zhou, and C. Yuan, “Chinese syntactic parsing algorithm based on segmentation of punctuation,” in *Journal of Chinese Information Processing*, vol. 21, no. 2, 3 2007.

[5] X. Li, C. Zong, and R. Hu, “A hierarchical parsing approach with punctuation processing for long Chinese sentences,” in *IJCNLP 2005: Companion Volume*, 2005.

[6] A. Ratnaparkhi, “A maximum entropy model for part-of-speech tagging,” in *Proceedings of EMNLP 1996*, 1996.

[7] B. Bohnet, “Efficient parsing of syntactic and semantic dependency structures,” in *Proceedings of CoNLL 2009: Shared Task*, 2009, pp. 67–72.

[8] A. Gesmundo, J. Henderson, P. Merlo, and I. Titov, “A latent variable model of synchronous syntactic-semantic parsing for multiple languages,” in *Proceedings of CoNLL 2009: Shared Task*, 2009, pp. 37–42.

[9] W. Che, Z. Li, Y. Li, Y. Guo, B. Qin, and T. Liu, “Multilingual dependency-based syntactic and semantic parsing,” in *Proceedings of CoNLL 2009: Shared Task*, 2009, pp. 49–54.