Zhijun Wu: Chinese Semantic Dependency Parsing with Third-Order Features

Zhijun Wu **Xuan Wang** **Xinxin Li**
Computer Application Research Center
School of Computer Science and Technology
Harbin Institute of Technology Shenzhen Graduate School
Shenzhen 518055, P.R.China
mattwu305@gmail.com wangxuan@insun.hit.edu.cn lixin2@gmail.com

Abstract

This paper presents our system participated on SemEval-2012 task: *Chinese Semantic Dependency Parsing*. Our system extends the second-order MST model by adding two third-order features. The two third-order features are grand-sibling and tri-sibling. In the decoding phase, we keep the k best results for each span. After using the selected third-order features, our system presently achieves LAS of 61.58% ignoring punctuation tokens which is 0.15% higher than the result of purely second-order model on the test dataset.

1 Introduction

Recently, semantic role labeling (SRL) has been a hot research topic. CoNLL shared tasks for joint parsing for syntactic and semantic dependencies both in the year 2008 and 2009, cf. (Surdeanu et al., 2008; Hajič et al., 2009; Bohnet, 2009). Same shared tasks in SemEval-2007 (Sameer S., 2007). The SRL is traditionally implemented as two subtasks, argument identification and classification. However, there are some problems for the semantic representation method used by the semantic role labeling. For example, the SRL only considers the predicate-argument relations and ignores the relations between a noun and its modifier, the meaning of semantic roles is related with special predicates. In order to overcome those problems, semantic dependency parsing (SDP) is introduced. Semantic dependencies express semantic links between predicates and arguments and represent relations between entities and events in text. The SDP is a kind of dependency parsing, and its task is to build a dependency structure for an input sentence and to label the semantic relation between a word and its head. However, semantic relations are different from syntactic relations, such as position independent. Table 1 shows the position independent of semantic relations for the sentence *XiaoMing hit XiaoBai with a book today*.

Today, XiaoMing hit XiaoBai with a book.
XiaoBai was hit by XiaoMing today with a book.
With a book, XiaoMing hit XiaoBai today.
XiaoMing hit XiaoBai with a book today.

Table 1: An example position dependency

Although semantic relations are different from syntactic relations, yet they are identical in the dependency tree. That means the methods used in syntactic dependency parsing can also be applied in SDP.

Two main approaches to syntactic dependency paring are *Maximum Spanning Tree* (MST) based dependency parsing and *Transition* based dependency parsing (Eisner, 1996; Nivre et al., 2004; McDonald and Pereira, 2006). The main idea of

MSTParser is to take dependency parsing as a problem of searching a maximum spanning tree (MST) in a directed graph (Dependency Tree). We see MSTParser a better chance to improve the parsing speed and MSTParser provides the state-of-the-art performance for both projective and non-projective tree banks. For the reasons above, we choose MSTParser as our *SemEval-2012* shared task participating system basic framework.

2 System Architecture

Our parser is based on the projective MSTParser using all the features described by (McDonald et al., 2006) as well as some third-order features described in the following sections. Semantic dependency paring is introduced in Section 3. We explain the reasons why we choose projective MSTParser in Section 4 which also contains the experiment result analysis in various conditions. Section 5 gives our conclusion and future work.

3 Semantic Dependency parsers

3.1 First-Order Model

Dependency tree parsing as the search for the maximum spanning tree in a directed graph was proposed by McDonald et al. (2005c). This formulation leads to efficient parsing algorithms for both projective and non-projective dependency trees with the Eisner algorithm (Eisner, 1996) and the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) respectively. The formulation works by defining in McDonald et al (2005a). The score of a dependency tree y for sentence x is

$$s(x, y) = \sum_{(i,j)\in y} s(i, j) = \sum w \cdot f(i, j)$$

f(i, j) is a multidimensional feature vector representation of the edge from node i to node j. We set the value of f(i, j) as 1 if there an edge from node i to node j. w is the corresponding weight vector between the two nodes that will be learned during training. Hence, finding a dependency tree with highest score is equivalent to finding a maximum spanning tree. Obviously, the scores are restricted to a single edge in the dependency tree, thus we call this first-order dependency parsing. This is a standard linear classifier. The features used in the first-order dependency parser are based on those listed in (Johansson, 2008). Table 2 shows the features we choose in the first-order parsing. We use some shorthand notations in order to simplify the feature representations: h is the abbreviation for head, d for dependent, s for nearby nodes (may not be siblings), f for form, le for the lemmas, pos for part-of-speech tags, dir for direction, dis for distance, '+1' and '-1' for right and left position respectively. Additional features are built by adding the direction and the distance plus the direction. The direction is left if the dependent is left to its head otherwise right. The distance is the number of words minus one between the head and the dependent in a certain sentence, if ≤ 5, 5 if > 5, 10 if > 10. ◎ means that previous part is built once and the additional part follow ◎ together with the previous part is built again.

Head and Dependent
h-f, h-l, d-pos ◎dir(h, d) ◎dis(h, d)
h-l, h-pos, d-f ◎dir(h, d) ◎dis(h, d)
h-pos, h-f, d-l ◎dir(h, d) ◎dis(h, d)
h-f, d-l, d-pos ◎dir(h, d) ◎dis(h, d)
h-f, d-f, d-l ◎dir(h, d) ◎dis(h, d)
h-f, h-l, d-f, d-l ◎dir(h, d) ◎dis(h, d)
h-f, h-l, d-f, d-pos ◎dir(h, d) ◎dis(h, d)
h-f, h-pos, d-f, d-pos ◎dir(h, d) ◎dis(h, d)
h-l, h-pos, d-l, d-pos ◎dir(h, d) ◎dis(h, d)
Dependent and Nearby
d-pos-1, d-pos, s-pos ◎dir(d, s) ◎dis(d, s)
d-pos-1, s-pos, s-pos+1 ◎dir(d, s) ◎dis(d, s)
d-pos-1, d-pos, s-pos+1 ◎dir(d, s) ◎dis(d, s)
d-pos, s-pos, s-pos+1 ◎dir(d, s) ◎dis(d, s)
d-pos, d-pos+1, s-pos-1 ◎dir(d, s) ◎dis(d, s)
d-pos-1, d-pos, s-pos-1 ◎dir(d, s) ◎dis(d, s)
d-pos, d-pos+1, s-pos ◎dir(d, s) ◎dis(d, s)
d-pos, s-pos-1, s-pos ◎dir(d, s) ◎dis(d, s)
d-pos+1, s-pos-1, s-pos ◎dir(d, s) ◎dis(d, s)
d-pos-1, d-pos, s-pos-1, s-pos ◎ dir(d, s) ◎ dis(d, s)
d-pos, d-pos+1, s-pos-1, s-pos ◎ dir(d, s) ◎ dis(d, s)
d-pos-1, d-pos, s-pos, s-pos+1 ◎ dir(d, s) ◎ dis(d, s)

Table 2: Selected features in first order parsing

3.2 Second-Order Model

A second order model proposed by McDonald (McDonald and Pereira, 2006) alleviates some of the first order factorization limitations. Because the first order parsing restricts scores to a single edge in a dependency tree, the procedure is sufficient. However, in the second order parsing scenario where more than one edge are considered by the parsing algorithm, combinations of two edges might be more accurate which will be described in the Section 4. The second-order parsing can be defined as below:

$$s(x,y) = \sum_{(i,j)\in y} s(i,k,j)$$

where k and j are adjacent, same-side children of i in the tree y. The shortcoming of this definition is that it restricts i on the same side of its sibling. In our system, we extend this restriction by adding the feature that as long as i is another child of k or j. In that case, i may be the child or grandchild of k or j which is shown in Figure 1.

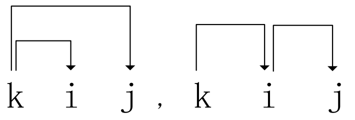

Figure 1: Sibling and grand-child relations.

Siblings
c1-pos, c2-pos◎dir(c1, c2)◎dis(c1, c2)
c1-f, c2-f◎dir(c1, c2)
c1-f, c2-pos◎dir(c1, c2)
c1-pos, c2-f◎dir(c1, c2)
Parent and Two Children
p-pos, c1-pos, c2-pos◎dir(c1, c2)◎dis(c1, c2)
p-f, c1-pos, c2-pos◎dir(c1, c2)◎dis(c1, c2)
p-f, c1-f, c2-pos◎dir(c1, c2) ◎dis(c1, c2)
p-f, c1-f, c2-f ◎dir(c1, c2) ◎dis(c1, c2)
p-pos, c1-f, c2-f◎dir(c1, c2) ◎dis(c1, c2)
p-pos, c1-f, c2-pos◎dir(c1, c2) ◎dis(c1, c2)
p-pos, c1-pos, c2-f◎dir(c1, c2) ◎dis(c1, c2)

Table 3: Selected features in second-order parsing

Shorthand notations are almost the same with the Section 3.1 except for that we use c1 and c2 to represent the two children and p for parent. In second-order parsing, the features selected are shown in Table 3. We divide the dependency distance into six parts which are 1 if > 1, 2 if > 2, … , 5 if > 5, 10 if > 10.

3.3 Third-Order Features

The order of parsing is defined according to the number of dependencies it contains (Koo and Collins, 2010). Collins classifies the third-order as two models, Model 1 is all grand-siblings, and Model 2 is grand-siblings and tri-siblings. A grand-sibling is a 4-tuple of indices (g, h, m, s) where g is grandfather. (h, m, s) is a sibling part and (g, h, m) is a grandchild part as well as (g, h, s). A tri-sibling part is also a 4-tuple of indices (h, m, s, t). Both (h, m, s) and (h, s, t) are siblings. Figure 2 clearly shows these relations.

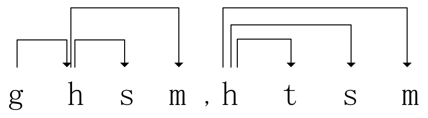

Figure 2: Grand-siblings and tri-siblings dependency.

Collins and Koo implement an efficient third-order dependency parsing algorithm, but still time consuming compared with the second-order (McDonald, 2006). For that reason, we only add third-order relation features into our system instead of implementing the third-order dependency parsing model. These features shown in Table 4 are grand-sibling and tri-sibling described above. Shorthand notations are almost the same with the Section 3.1 and 3.2 except that we use c3 for the third sibling and g represent the grandfather. We attempt to add features of words form and parts-of-speech as well as directions into our system, which is used both in first-order and second-order as features, but result shows that these decrease the system performance.

Tri-Sibling
c1-pos, c2-pos, c3-pos◎dir(c1, c2)
Grandfather and Two Children
g-pos, c1-pos, c2-pos◎dir(c1, c2)
g-pos, p-pos, c1-pos, c2-pos◎dir(c1, c2)

Table 4: Third-order features.

4 Experiment result analysis

As we all know that projective dependency parsing using edge based factorization can be processed by the Einster algorithm (Einster, 1996). The corpus given by SemEval-2012 is consists of 10000 sentences converting into dependency structures from Chinese Penn Treebank randomly. We find that none of non-projective sentence existing by testing the 8301 sentences in training data. For this reason, we set the MSTParser into projective parsing mode.

We perform a number of experiments where we compare the first-order, second-order and second-order by adding third-order features proposed in the previous sections. We train the model on the full training set which contains 8301 sentences totally. We use 10 training iterations and projective decoding in the experiments. Experimental results show that 10 training iterations are better than others. After adjusting the features of third-order, our best result reaches the labeled attachment score of 62.48% on the developing dataset which ignores punctuation. We submitted our currently best result to SemEval-2012 which is 61.58% on the test dataset. The results in Table 5 show that by adding third-order features to second-order model, we improve the dependency parsing accuracies by 1.21% comparing to first-order model and 0.15% comparing to second-order model.

Models	LAS	UAS
First-Order	61.26	80.18
Second-Order	62.33	81.40
Second-Order+	62.48	81.43

Table 5: Experimental results. Second-Order+ means second-order model by adding third-order features. Results are tested under the developping dataset which contains the heads and semantic relations given by organizer.

5 Conclusion and Future Work

In this paper, we have presented the semantic dependency parsing and shown it works on the first-order model, second-order model and second-order model by adding third-order features. Our experimental results show more significant improvements than the conventional approaches of third-order model.

In the future, we firstly plan to implement the third-order model by adding higher-order features, such as forth-order features. We have found that both in the first-order and second-order model of MSTParser, words form and lemmas are recognized as two different features. These features are essential in languages that have different grid, however, which are the same in Chinese in the given dataset. Things are the same in POS (part-of-speech tags) and CPOS (fine-grid POS) which are viewed as different features. For the applications of syntactic and semantic parsing, the parsing time and memory footprint are very important. Therefore, secondly, we decide to remove these repeated features in order to reduce to training time as well as the space if it does not lower the system performance.

Acknowledgments

The authors would like to thank the reviewers for their helpful comments.

References

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Márquez, and JoakimNivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic andsemantic dependencies. In *Proceedings of the 12th CoNLL-2008*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antonia Martí, Llu'is Márquez, Adam Meyers, Joakim Nivre, SebastianPado, Jan Štepánek, Pavel Stranák, Miahi Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the 13th CoNLL-2009, June 4-5*, Boulder, Colorado, USA.

Bohnet, Bernd. 2009. Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of CoNLL-09*.

Ryan McDonald. 2006. Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing. Ph.D. thesis, University of Pennsylvania.

Ryan McDonald and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algrithms. In *In Proc. of EACL*, pages 81–88.

Ryan. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of the 43rd Annual Meeting of the ACL*.

Ryan. McDonald, F. Pereira, K. Ribarov, and J. Hajiˇc. 2005c. Non-projective dependency parsing using spanning tree algorithms. In *Proc. HLT-EMNLP*.

Richard Johansson. 2008. *Dependency-based Semantic Analysis of Natural-language Text*. Ph.D. thesis, Lund University.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhaen.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-Based Dependency Parsing. In *Proceedings of the 8th CoNLL*, pages 49–56, Boston, Massachusetts.

Terry Koo, Michael Collins. 2010. Efficient Third-order Dependency Parsers. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11.