Peking: Building Semantic Dependency Graphs with a Hybrid Parser

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun* and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{duyantao,zhangxunah,ws,wanxiaojun}@pku.edu.cn
zhangf717@gmail.com

Abstract

This paper is a description of our system for SemEval-2015 Task 18: Broad-Coverage Semantic Dependency Parsing. We implement a hybrid parser which benefits from both transition-based and graph-based parsing approaches. In particular, the tree approximation method is explored to take advantage of well-studied tree parsing techniques. Evaluation on multilingual data sets demonstrates that considerably good semantic analysis can be automatically built by applying state-of-the-art data-driven parsing techniques.

1 Introduction

Dependency grammar is a long-standing tradition that determines syntacto-semantic structures on the basis of word-to-word connections. It names a family of approaches to linguistic analysis that all share a commitment to typed relations between ordered pairs of words. Partially due to the powerful expressiveness of bi-lexical dependency structures, the corresponding parsing problem has been widely studied especially in the last decade. The majority of these studies, however, only focus on tree-structured representations. Beyond tree-shaped structures, SemEval-2014 Task 8 (Oepen et al., 2014) seeked to stimulate the dependency parsing community to move towards more general graph processing. Quite a number of teams all over the world participated in this shared task, which suggests a growing community interest in parsing into graph-shaped dependency representations.

*Email correspondence.

SemEval-2015 Task 18 is a subsequent task of SemEval-2014 Task 8. Following several well-established syntactic theories, this task proposes using graphs to represent semantics and provides high-quality annotations for three typologically different languages. We have developed a system, dubbed DZSW14 (Du et al., 2014) for the task last year. The system employed a hybrid architecture which benefits from both transition-based and graph-based parsing approaches. Evaluation on multiple English data sets provided by SemEval-2014 indicated that DZSW14 is able to obtain high-quality parsing results. Following the key idea to employ heterogeneous models to enhance hybrid parsing, we extend DZSW14 by developing more tree approximation models, namely the weighted tree approximation models. Evaluation on multilingual data sets provided by this year's task confirms the effectiveness of the techniques we have studied.

In this paper, we first give an introduction of the architecture of the baseline system DZSW14. Then we demonstrate the weighted tree approximation models. Finally we show the experiment results on SemEval-2015 Task 18. The tree approximation system can be downloaded at `http://www.icst.pku.edu.cn/lcwm/grass`.

2 Baseline System: DZSW14

Our system is based on the system we constructed for SemEval-2014 Task 8. In this section we present a brief overview of its architecture. Refer to (Du et al., 2014) for more information.

Inspired by the research on discriminative dependency tree parsing, DZSW14 employed a hybrid parsing architecture. DZSW14 explored two

Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 927–931,
Denver, Colorado, June 4-5, 2015.

kinds of heterogeneous approaches: transition-based and tree approximation approaches. The transition-based model use transitions on configurations to obtain graph parses, while the tree approximation model transform graphs into trees for training and test. To further combine the complementary prediction power, DZWS14 applied a voting-based ensemble method.

2.1 Transition-Based Models

Transition-based models consist of transitions and configurations that can be manipulated by the transitions. The configurations generally encode the information of the current parsing state, especially including partial parsing results, and the transitions can be applied to a configuration, turning it into a new one. When the system reaches any acceptable configuration, a coherent semantic graph is also successfully built. The key to the success of building transition-based parsers is to train good classifiers to approximate transition oracles. DZSW14 implements 5 different transition systems for graph parsing. Experiments from last year's evaluation suggest that this method can be applied to build considerably good parsers for more general linguistic graphs.

2.2 Tree Approximation Models

The core of tree approximation is transformations between graphs and trees. At the training time, we convert the dependency graphs from the training data into dependency trees, and train second-order arc-factored models[1] (Bohnet, 2010). At the test phase, we parse sentences using this tree parser, and convert the output trees back into semantic graphs. In DZSW14, We develop several different methods to convert a semantic graph into a tree. The main idea is to apply graph traversal algorithms to convert a directed graph to a directed tree. During the traversal, we may lose or modify some dependency relations in order to make a tree.

2.3 Experience from DZSW14

From a lot of experiments on DZSW14, we learned several lessons as follows.

- Overall, the tree approximation models perform better than the transition-based models.
- The outputs of the different models exhibit significant diversity.
- The model ensemble is quite effective, resulting in a boost in performance.

This motivates us to explore more heterogeneous tree approximation models for this year's evaluation.

3 Weighted Tree Approximation Models

In our system for SemEval-2015, we develop more tree approximation models for model ensemble. We call the graph-to-tree conversions in DZSW14 unweighted conversions since every edge in the graph are treated equally. In this section, we demonstrate weighted conversions which assign weights for different edges.

3.1 Weighted Conversion

Given a graph $G = \langle V, E \rangle$, the edge selection in the unweighted conversion is locally decided by its current traversal state. In the weighted conversion, we take the importance of different edges into account and try to globally improve the integrity with respect to the losing edges. For example, in the top of Figure 1, the undirected edges $(\text{Ward}, \text{was})$, $(\text{Ward}, \text{relieved})$, $(\text{was}, \text{relieved})$ form a cycle. Only two edges can be kept by the converted tree T. It allows us to decide which edges to keep according to the sum of the weights of them.

Let $x \rightarrow_t y$ denote edges in the tree, and $x \rightarrow_g y$ edges in the graph. We assign each possible edge $x \rightarrow_t y$ a heuristic value $\omega(x, y)$, and intend to obtain a tree with maximum weight . More formally, the result $T^{\max} = (V, E_t^{\max})$ contains the maximum sum of values of edges:

$$T^{\max} = \arg\max_{T=(V,E_t)} \sum_{(x,y)\in E_t} \omega(x, y)$$

3.2 Weight

We define weight $\omega(x, y)$ as follows, where I is the indicator function:

- $\omega(x, y) = A(x, y) + B(x, y) + C(x, y)$: The weight is separate into 3 parts.

[1] The mate parser (`code.google.com/p/mate-tools/`) is used.

- $A(x, y) = I_{x \to y \in E \vee y \to x \in E}(x, y) \times a$: a is the weight for the existing edge on graph ignoring direction.
- $B(x, y) = I_{x \to y \in E}(x, y) \times b$: b is the weight for the directed edge in the graph.
- $C(x, y) = n - |x - y|$: This is to value the importance of edges where n is the length of sentence. We consider edges linking closer words more important because they are generally easier to be predicted.
- $a \gg b \gg n$ or $a > b \times n > n \times n$: First the transformed tree should contain the original edges in G as many as possible. Then we need to consider the quantity of edges with correct direction in G. And the distance between nodes in the sentence is in the last place.

3.3 Decoding

After the edges are weighted, the core decoding task for graph transformation can be solved by maximum spanning tree (MST) algorithms, where the search space $\mathcal{T}$ consists of all projective and non-projective dependency trees. To transform a graph to a projective tree, we use Eisner's algorithm, and for non-projective, we use Chu-Liu-Edmonds algorithm.

3.4 Adding Labels

Now we get the MST $T^{\max}(V, E_t^{\max})$. For each $(x, y) \in E_t^{\max}$, we assign a new label to (x, y) as follows,

Case 1: $x \to_g y$, add the original label in $G(V, E)$ to the new edge $x \to_t y$;

Case 2: $y \to_g x$, add the original label with symbol $\tilde{\text{R}}$ to $x \to_t y$;

Case 3: $x \to_g y \wedge y \to_g x$, add label as Case 1;

Case 4: $x \not\to_g y \wedge y \not\to x$, add label *None* to the edge $x \to_t y$.

To improve the coverage of original edges, a variant model with modified labels in trees to help encode more edges in graphs.Suppose that $x \to y$ is a lost edge which is not on the new dependency tree but is on the original dependency graph. The statistic shows the structure of a majority of lost edges are in one of three different types:

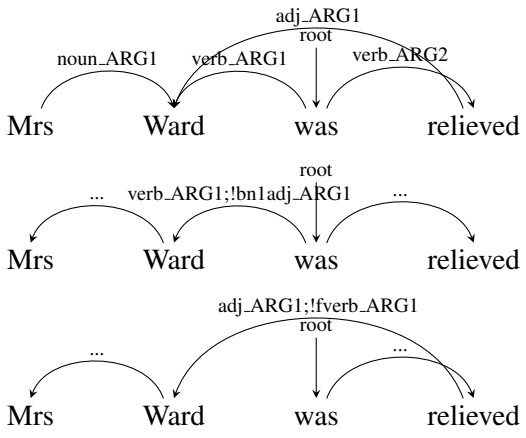

Figure 1: One dependency graph and two possible dependency trees after converting.

1. The nodes are siblings.
2. One is the grandparent of the other.
3. One is the great-grandparent of the other.

The conversions can be enhanced by adding more symbols to labels to indicate lost edges if they are of the three types above. The method is to append semicolon (;) and exclamation mark (!) to some degree and then add new label with information of lost edge directly. If $x \to_g y$ is not in the converted tree and its structure is of one of aforementioned types, we change the label connecting node y and its parent node with assumption that y is not higher than x in the dependency tree.

- x is great-grandparent node of y: New label is the label of $x \to_g y$ following the symbol 'g'.
- x is grandparent node of y: New label is the label of $x \to_g y$ following the symbol 'f'.
- x and y are siblings: Let z be the two nodes' parent. We sort all the z's children by the order of position in the sentence And we use an integer P to indicate the position. If the two siblings are on the same side of z, P will be the distance of the two siblings' positions in the sorted children sequence and extra symbol will be 'y'. If the two siblings are on the different sides of z, extra symbol will be 'n' and P will be x's rank in the same side's nodes in the sorted children sequence. New label is the

symbol 'b' with extra symbol followed and the label of $x \rightarrow_g y$.

If node y is higher than x in the dependency tree, we would add symbol $\tilde{\text{R}}$ to indicate the additional edge is reversed. Figure 1 is an example of a converted tree.

4 Model Ensemble

We select 9 tree approximation models from DZSW14, and propose 4 new weighted models ({projective, non-projective} $\times$ {original, label-modification-variant}). Together with the transition models, we have to combine the outputs of them into one. We use a simple voter to combine the outputs just like in DZSW14. For each pair of words of a sentence, we count the number of the models that give positive predictions. If the number is greater than a threshold, we put this arc to the final graph, and label the arc with the most common label of what the models give.

Furthermore, we find that the performance of the tree approximation models are better than the transition based models, so we assign weights for individual models too. Then instead of just counting, we sum the weights of the models that give positive predictions. The tree approximation models are assigned higher weights.

5 Sense Labeling

In this task, two representations DM and PSD of English require to label the words additional sense label. We develop a sequence labeler for this requirement. The sequence labeler is based on a second-order linear-chain global linear model and utilize the perceptron algorithm for parameter estimation. To accelerate processing, we apply a Viterbi decoder but constrain it with beam search. In particular, the number of cells in the dynamic programming table for each word is bounded by a fixed beam size. This decoder can be also viewed as a beam decoder with state-merging.

The representation DM can be labeled directly. However due to the large amount of different senses in representation PSD, it is difficult to label senses without preprocessing. We finally decide to filter out the rare senses that have a frequency lower than 10, substituting "unknown" for them.

Algorithm	DM_{en}	PAS_{en}	PSD_{en}	PAS_{cs}	PSD_{cz}
PROJ	4.24	6.31	8.89	9.36	3.56
NON-PROJ	2.31	6.16	8.42	9.04	2.81
PROJ′	2.30	1.85	2.73	3.26	2.21
NON-PROJ′	0.60	1.62	2.33	3.07	1.55

Table 1: Edge loss of conversion algorithms (%).

Domain	Format	LP	LR	LF	LM
id	DM_{en}	0.9093	0.8732	0.8909	0.2702
	PAS_{en}	0.9290	0.8967	0.9126	0.3028
	PSD_{en}	0.7860	0.7293	0.7566	0.0872
	PAS_{cs}	0.8191	0.7434	0.7794	0.1144
	PSD_{cz}	0.8475	0.8215	0.8343	0.2809
ood	DM_{en}	0.8429	0.7953	0.8184	0.2499
	PAS_{en}	0.8947	0.8510	0.8723	0.3012
	PSD_{en}	0.7736	0.6961	0.7328	0.1790
	PAS_{cs}	0.6941	0.6002	0.6437	0.1146

Table 2: Final results of the ensembled model.

6 Experiments

We participated in the closed track. The tree approximation algorithms may cause some edge loss, and the statistics for the weighted conversions are shown in Table 1. We can see that all the algorithms cause edge loss, and edge loss of the variants is much lower. In addition, non-projective tree conversions cause less loss compared to projective tree conversions. Edge loss may result in a lower recall and higher precision, but we can tune the final results during model ensemble.

The final results given by the organizers are shown in Table 2. Here we only give the labeled score.

7 Conclusion

Based on our previous system DZSW14, we developed a hybrid system for SemEval-2015 Task 18. Our new system extends DZSW14 by providing several more tree approximation models. The final result shows that our system as well as our new models are effective.

Acknowledgement

The work was supported by NSFC (61300064, 61170166 and 61331011) and National High-Tech R&D Program (2012AA011101).

References

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August. Coling 2010 Organizing Committee.

Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, Dublin, Ireland.