Team Szeged @ EPE 2017 shared task - First experiments in a generalized syntactic parsing framework

Zsolt Szantó, Richárd Farkas
University of Szeged, Institute of Informatics
{szantozs, rfarkas}@inf.u-szeged.hu

Abstract

In this article, we describe the submissions of Team Szeged for the EPE 2017 shared task. We introduce three approaches and first experiments to exploit the opportunities of the general dependency graph representation of the shared task.

1 Introduction

The goal of the First Shared Task on Extrinsic Parser Evaluation (EPE 2017) was to estimate "the relative utility of different types of dependency representations for a variety of downstream applications that depend heavily on the analysis of grammatical structure."

To enable different types of dependency representations, the organizers of the shared task introduced a very general graph-based representation of 'relational' structure reflecting syntactic-semantic analysis. The nodes of this graph correspond to lexical units, and its edges represent labeled directed relations between two nodes. Nodes can be defined in any terms of (in principle arbitrary) sub-strings of the surface form of the input sentence. This representation allows overlapping and empty (i.e. zero-span) node sub-strings as well. Moreover, nodes and edges are labeled by attributevalue maps without any restriction on the attribute set.

This very general graph-based representation opens brand new ways for expressing syntactic or semantic information besides the standard dependency tree formalism. We understood the call of the shared task in a generalized way and came up with ideas which aim to leverage the opportunities of the general representation beyond dependency parse trees. We experimented with couple of such ideas (instead of trying to achieve high scores in the shared task) and we shall introduce them in this paper.

The contribution of this work consists of three independent pack of experiments in the EPE 2017 setting. We emphasize that these are only first experimental results and there is a plenty of space for analyzing these approaches and also to come up with other ideas leveraging the broadly understood task specification.

In the first set of experiments (§2), we start from the classic dependency parsing approach but instead of a single dependency parse we express the distribution of possible dependency parses given a sentence in the graph-based general representation. In Section 3, we introduce a possible solution for enriching the dependency parse by constituent information given by a standard phrase-structure parser. In this way, various syntactic representations can be represented in the graph and information is not lost because the downstream application can only accept a single dependency parse tree. Furthermore, in the EPE 2017 setting we can send a blended relational structure to the downstream task, like a parse distribution and blended version of different syntactic approaches, and the downstream application is able to machine learn which type of syntactic structure or phenome or even which combination of syntactic information is useful for itself.

Our last batch of experiment (§4) is a consequence of this objective, i.e. the relational representation has to be useful for the downstream application. Here, we tried to automatically recognize which dependency parse labels are useful to a downstream task and collapsed the useless ones.

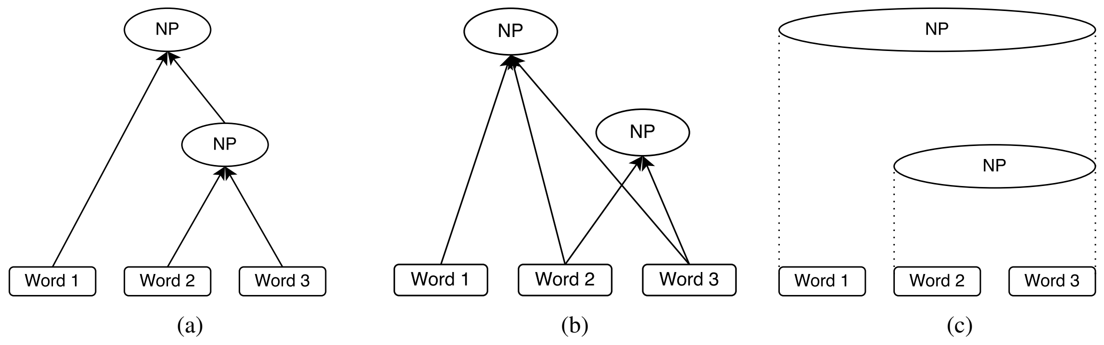

Figure 1: Alternatives for representing constituent trees in the general dependecy graph format.

2 Parse Distribution as Input for Downstream Applications

Standard dependency parsers output a single dependency parse tree. Our hypothesis was that a downstream application could profit from having access to the distribution of possible parses and not just to the most likely parse tree. The distribution over possible parses estimated by the parsing model might be useful for a downstream application because it might reveal that edges or their labels are less confident or also point out relatively highly probably dependencies which are not part of the best single parse tree. The general graph-based representation of EPE 2017 enables to express the distribution over possible edges and labels, i.e. possible parses.

Getting out the density estimation from a particular parser is usually complicated because of both theoretical and practical (software implementation) issues. Hence we decided to use an approximation of edge and label likelihoods based on top-K parses for our first experiments. Our assumption here is that the top K output of a parser model contains most of the useful bi-lexical dependencies and the frequency of a particular dependency counted among the K parses is a good enough approximation for its likelihood (this idea is similar to constituent-level strategy of the Berkeley product parser (Petrov, 2010)).

We added each edges from the k-best trees of a parser to the general dependency graph. We also added a new label to all edges whose value is the frequency of the same edge label pairs among the k parses. For these experiments we used the MST-Parser (McDonald et al., 2005) what we trained on Universal Dependencies v2 (Nivre et al., 2015) and we asked for the 10-best trees with default parameters.

3 Constituents in the (Bi-Lexical) Relational Representation

Constituency parsers focus on the phrases/constituents and phrase structure of the sentence, i.e. follow a non bi-lexical syntactic representation. Several applications might prefer bi-lexical representations (like the ones based on predicate-argument structures) while others might prefer constituency (like scope detection). Fortunately, the general graph representation of EPE 2017 enables us to put both the dependency and constituency parse output into a blended syntactic graph. Hence we do not have to choose between the two approaches but the downstream application can machine learn which syntactic phenomena is useful for itself or even can learn patterns in the graph consisting of information from both constituency and dependency. Couple of previous work has shown that the two syntactic representation and their parsers can work together efficiently cf. (Farkas and Bohnet, 2012). We believe that is especially true for using them jointly in downstream applications.

There are many possible ways how we can represent a constituent tree in the general dependency graph format. Although these representations contain the same information because of the feature extractors of the downstream applications they can have different effect in practice.

An interesting opportunity of the EPE 2017 general graph representaton is that it enables the creation of virtual nodes. This feature gives the posibility to create a new node for each non-terminal in a constituent tree. Our three proposal differs in the way these virtual nodes are linked to the overt nodes in the graph.

1. In the first setup, shown by the Figure 1a, we connect each of the children to their direct

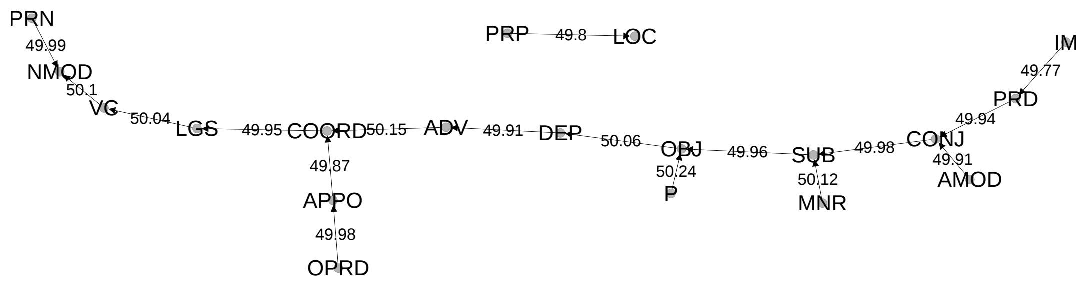

Figure 2: Label adjustment graph.

parents. In this way our graphs will be very similar to a constituent tree. In this example the Word 2 and Word 3 are connected to an NP, that NP and Word 1 is connected to the another NP.

2. Another posibility is when each of the nodes are connected to all ancestor non-terminals (Figure 1b). In that case there is a direct relation between a constituent and their descendants. In the current example the higher level NP directly contains the children (Word 2, Word 3) of its child. This representation has the hope that the feature extractor of downstream applications can directly generate features about the ancestors without recursive rules

3. A different approach is where we give the covering area for each new non-terminal (Figure 1c). In these cases like in the previous we have not got direct information about the connection between the nonterminals. On the other hand, it can help for an application which uses the position of a node.

For constituent parsing we used the Berkeley Parser (Slav Petrov and Klein, 2006) with default parameters and pretrained model (eng-sm6). In our submission we used the second and third methods in the dependency graph format.

4 Label Set Adjustment Driven by Downstream Applications

Different downstream applications might utilize different type of grammatical patterns. The simplest case is that a downstream application might extract important features from particular edge labels while features over other edge labels are negligible in its machine learnt model. Moreover, different applications might utilize different type of dependencies, see for example event recognition versus negation scope detection.

We propose a simple procedure to recognize edge labels which can be collapsed into other edge labels because their discrimination does not give any added value to the downstream application in question. We start from the full set of edge labels and systematically check what is the effect of collapsing two particular labels evaluated through the downstream application.

We calculated for all label pairs what is going to happen if we replace one dependency label to another. For our experiments we used the TEES system, but because we did not have enough time to retrain the TEES system for all combination, we trained it once with the full label set and we did the prediction part separatly to each dependency label pairs. In this prediction part we replaced each of the labels with each of another labels on the full development set. We got a complete directed graph where the nodes are the labels and edges contains the scores from the TEES system with the merged labels. For each node we kept the outcoming edge with maximum weight i.e. when the replace was the most efficient. When there were two edges between two node we removed the smaller.

Figure 2 contains the graph we got. (When we ran the TEES system with default parameters we got 49.76 with original labels). By using this graph we started replacing the nodes from the highest edge weight to the lowest. We evaluated the new labelset in every step and we found the best result after three steps, 50.36, which is slightly better than the best merged pair. After that we did the three replace steps in the full dataset. Because of lack of time, we cannot make the replacement in the full dataset and retrain the TEES before the submission deadline. We sent the trees with collapsed labelset. We found lower scores than the baseline on the TEES test set. We also

	Event Extraction	Negation Resolution	Opinion Analysis
mate - baseline	47.84	61.98	65.87
mate + label adjustment	47.37	60.53	66.33
mate + constituent	46.71	61.26	63.13
mate + mst - baseline	46.69	59.78	63.25
mate + mst - k-best	45.96	59.05	62.5

Table 1: Final result in evalutaion set.

could not use this method for the other downstream applications.

5 Results

The Table 1 shows our official results achived on the shared task. The *baseline - mate* is one of our baselines where we just run the mate parser (Bohnet, 2010) with pretrained model. The second and third rows contain the result of *label adjustment* and *constituent parsing* experiments. The fourth row contains another baseline when we applied the MSTParser and the last row is shows the scores of our *k-best experiment* (we used MSTParser here).

Unfortunatelly, the deeper analysis of results is left for future work. For example, the reason why the *combination of k-best parse* get lower result in every task than the *baseline-MST* is maybe the feature extractors were prepared for dependency representations and not for distribution graphs.

5.1 Event Extraction

In the event extraction task we can not beat our baselines, all of our modifications – including the label adjustment which is optimized for this task – get negative effect. The dependency label merging mechanism what we directly developed on this task also failed.

5.2 Negation Resolution

One of the main motivation of the constituent based approach was the negation resolution task. The scope of the negations are usually a close what can we identify with constituent parsers. This *constituency-based system* got better result in three out of four scope-focused evaluation metric than our baseline. Table 2 shows the detailed comparison of the baseline and the constituent system.

The following example shows how can the constituent parse help:

"I join in it because there is ***no*** *other way in the world by which justice can be gained."*

	baseline		mate + const	
	dev	test	dev	test
Scope Match	78.42	80.00	77.98	81.14
Scope Tokens	86.64	89.17	87.38	89.27
Event Match	75.47	67.90	72.90	65.20
Full Negation	62.15	61.98	59.91	61.26

Table 2: Detailed results of the *baseline - mate* and the *mate + constituent* systems in negation resolution task.

The scope of the *no* negation clue starts from the *there* and end with the *gained* word. Our baseline system marked the negation from the *there* to the *justice*, but the constituent based method found the correct scope. If we look at the constituent tree we see the full scope is covered by a constituent with *S* label. Instead of scope detection the constituent based information can't help in the event detection subtask.

5.3 Opinion Analysis

In the opinion analysis task the *label adjustment* method imporved 0.5 percetage point against the *mate-baseline* and got the best results in the shared task in Holders (In Vitro) metric. It seems the label collapsions what our method found in the event extraction task is more general than we expected. On the other hand, it is still an open question why this label collapsion did not work at the event extraction task's evalaution set.

6 Conclusions

We introduced the contribution of team Szeged to the EPE 2017 Shared Task. We proposed three approaches for relational representation of syntax beyond the canonical dependency parse tree approach. Although these experiments are only the very first tries for such representation we hope that this might give ideas about the important topic of syntactic and semantic parser solutions which aim to be (automatically) fine-tuned for a particular downstream applications demands.

Acknowledgments

We thank to the organizers of the EPE shared task their hard work on extending the feature extractor of downstream applications according to our weired ideas.

The research work of Richrd Farkas has been supported by the Jnos Bolyai scholarship of the Hungarian Academy of Sciences.

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*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '10, pages 89–97.

Richárd Farkas and Bernd Bohnet. 2012. Stacking of dependency and phrase structure parsers. In *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, Mumbai, India, pages 849–866.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '05, pages 523–530.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, et al. 2015. Universal dependencies 1.2 .

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '10, pages 19–27.

Leon Barrett Romain Thibaux Slav Petrov and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*. Association for Computational Linguistics, Sydney, Australia, pages 433–440.