Turku: Semantic Dependency Parsing as a Sequence Classification

Jenna Kanerva[1,2], **Juhani Luotolahti**[1], **Filip Ginter**[1]
[1] Department of Information Technology, University of Turku, Finland
[2] University of Turku Graduate School (UTUGS), Turku, Finland
jmnybl,mjluot,figint@utu.fi

Abstract

This paper presents the University of Turku entry to the SemEval-2015 task on Broad-Coverage Semantic Dependency Parsing. The system uses an existing transition-based parser as a sequence classifier to jointly predict all arguments of one candidate predicate at a time. Compared to our 2014 entry, the 2015 system gains about 3pp in terms of F-score for a fraction of the development time. Depending on the subtask, the difference between our entry and the winning system ranges between 1 and 5pp.

1 Introduction

The SemEval-2015 task on Broad-Coverage Semantic Dependency Parsing is a continuation to the semantic parsing shared task organized for the first time in 2014. The objective of the shared task is to produce a rich semantic analysis for a given sentence in three distinct annotation formats. In contrast to the 2014 task, this year predicate disambiguation and two additional languages are included: Czech data from the Prague Czech–English Dependency Treebank (Hajič et al., 2012) and Chinese data from the Penn Chinese Treebank (Xue et al., 2005). For English and Czech also out-of-domain test data is provided in order to test the generalization ability of the systems.

The semantic parsing task includes three different tracks. In the closed track the systems must be trained using only the official training data, whereas in the open track all additional sources of information are allowed. Together with the training data the organizers provided also syntactic dependency parses produced in the Stanford Dependencies scheme (De Marneffe and Manning, 2008) with the dependency parser of Bohnet and Nivre (2012). In addition to the closed and open tracks, also a gold track is included, where gold standard dependency parses are given for both training and test data.

This paper describes our system used to take part in the open and gold tracks of the shared task. The system is a sequence classifier built on top of an existing dependency parser. The main idea behind the implementation is to turn the task of predicting all arguments for a single predicate to a sequence classification problem, but still process each predicate independently. Predicting one predicate at a time feels very natural when working with data annotated in PropBank style (Palmer et al., 2005), and since our main objective is to develop an SRL system optimized for Finnish PropBank (Haverinen et al., 2013), we did not want to merely follow the main methods from last year. Our system also requires syntactic analyses of the data, which is why we participated only on the tasks which allow their use (open and gold tracks). The system will be described in detail in Section 3.

2 Related Work

The main approaches in the 2014 semantic dependency parsing task (Oepen et al., 2014) relied on the methods developed in the context of syntactic parsing, and existing state-of-the-art dependency parsers were widely used. Systems using dependency parsers are mainly based on graph-to-tree transformations (Koller, 2014; Schluter et al., 2014), parsers

Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 965–969,
Denver, Colorado, June 4-5, 2015.

able to produce directed acyclic graphs (Ribeyre et al., 2014; Kuhlmann, 2014), or a combination of these two (Du et al., 2014). The winner system of the 2014 open track is based on the graph-based dependency parser able to produce full non-projective graphs (Martins and Almeida, 2014).

The system we used to participate in the same task last year was a pipeline of three different support vector machine classifiers trained separately for dependency detection, role assignment and top node prediction, where each governor–dependent pair was classified individually without any global view of the semantic structures (Kanerva et al., 2014a). A similar approach with the exception of using a structured support vector machine and therefore gaining a bit more of a global view to the problem was introduced by Jeffrey et al. (2014).

3 System Architecture

The main approach is based on the recent progress on syntactic dependency parsing, yet taking a completely different approach than the mainstream graph-to-tree transformation methods and DAG parsers discussed in Section 2. Our main focus is to process each predicate independently (i.e. other predicates and their arguments do not affect the decision), but when assigning arguments for one predicate, keep a global view of arguments already predicted for this particular predicate.

The system is built on top of the open-source Turku transition-based dependency parser[1] to obtain the full functionality of such a parser and to be able to freely modify it to fit to the needs of our approach. The Turku Dependency Parser is an implementation of the parser of Bohnet and Kuhn (2012), with full functionality of that parser, including e.g. online learning implemented with the generalized perceptron (Collins, 2002), beam search and graph-based completion features, and the full feature representation taken from the Bohnet and Nivre (2012).

3.1 Data Processing

Before training the parser, the data is processed to meet the requirements of the standard, off-the-shelf dependency parsers. As the arguments are predicted separately for each predicate, semantic graphs can be subtracted into several smaller units where each subgraph preserves the semantic arguments of one particular predicate. This means that each sentence is turned into as many pseudotrees as there are tokens in the sentence, where each token in turns acts as a candidate predicate and preserves only its own arguments and all other relations are dropped from this particular pseudotree. These pseudotrees are finalized by attaching all other tokens to the candidate predicate with an empty relation type *NOTARG*, which at the same time causes the candidate predicate to be the root token of the tree. Since the data does not include self-loops or multiple arguments between the same governor and dependent pair, this transformation can be made without losing any information. These pseudotrees created from one sentence can again be merged into a one semantic graph by just preserving the real semantic relations and leaving out the empty *NOTARG* relations.

As will be explained later, syntactic parses are a major source of features. For English, the syntactic parses are obtained from the companion and gold data provided by the organizers. Since for Czech and Chinese no companion data was available, the Czech syntactic representation is obtained using the MaltParser (Nivre et al., 2007) trained on the training section of the Prague Dependency Treebank (Hajič et al., 2000) and the Chinese analysis is acquired using DuDuPlus, a graph-based dependency parser (Chen et al., 2009) with a model trained on the training section of the Chinese Treebank (Xue et al., 2005).

3.2 Transition System

Since the structure of the input trees is completely flat (i.e. all words are attached to the sentence root, which is the predicate under inspection) the transition system of the parser can be simplified substantially. For every token other than the root, only the relation type must be predicted (using the *NOTARG* relation for tokens which are not arguments of this particular predicate). Thus, the transition system is modified to keep the root token always in the parsing stack, and one by one taking the next token from the queue, predicting its relation type and reducing it from the stack in a single operation.

Since the simplified transition system requires that the root token is in the stack already when the

[1] https://github.com/jmnybl/Turku-Dependency-Parser

parsing starts, the order of the tokens in the parsing queue must be manipulated. Manipulating the parsing queue also changes the order in which the predictions are made and since the parser is beam searched and the system has an ability to recover from a wrong prediction made earlier on, the different order of predictions may affect the final sequence of predictions. Two different approaches are tested. First and by far the simplest method is to use the normal linear order of the tokens and just remove the sentence root from the queue (run 1 in the official results). Second method is to reorder the tokens based on the syntactic distance, where the tokens closest to sentence root in the syntactic tree are first in the queue (and thus their relations are also predicted first) (run 2 in the official results). The idea behind this is to assume that tokens which are most likely to be arguments of the predicate are close to it in the syntactic representation and therefore predicted first. Official results showed that the first method performed better and therefore all numbers reported in this paper are based on the first run.

3.3 Feature Generation

The basic features used are based on the standard features of dependency parsers. However, few modifications to the parser feature representation were made. The function of the graph-based completion features is to model the partial structures of the tree already built at any given point. Since in the simplified transition system all tokens are forced to be dependents of the sentence root, taking into account all created relations would not distinguish semantically meaningful tokens from all other tokens (as is in the case of syntactic parsing where only real syntactic dependents are attached to any given token). Thus, tokens attached with the empty relation *NOTARG* are discarded and the graph-based completion features are created only from the real semantic relations.

Additional features are created from the syntactic structure of the sentence. The most important feature extracted from the syntactic tree is the path between the predicate and the potential argument. Two variations of the syntactic path are used; the dependency types and the part-of-speech tags between the two tokens. If the distance of the tokens is smaller than seven dependencies, full paths are used as features, otherwise only the beginning and the end of the path are used. Finally, from each aforementioned path all dependency type and part-of-speech tag trigrams are created.

	P	R	F	UF
Open in-domain				
DM	87.80	84.60	86.17	88.07
PAS	91.38	89.87	90.62	91.91
PSD	76.10	71.32	73.63	86.44
Overall	85.09	81.93	83.47	88.81
Open out-of-domain				
DM	81.54	76.63	79.01	81.68
PAS	86.95	84.98	85.95	87.83
PSD	74.92	68.55	71.59	86.54
Overall	81.14	76.72	78.85	85.35

Table 1: English open track (in-domain & out-of-domain) results in terms of precision (P), recall (R), labeled F-score (F) and unlabeled F-score (UF).

3.4 Top Node and Sense Prediction

Prediction of top nodes and predicate senses are implemented as separate steps and carried out after the argument prediction. The top nodes are predicted in the same manner as in our 2014 system (Kanerva et al., 2014a), where a support vector classifier is trained to classify individual tokens.

Predicate senses are predicted with the approach introduced by Kanerva and Ginter (2014), where vector space representations of tokens are used to calculate an average vector to represent each individual sense. Then for each predicate the sense is assigned by calculating a vector to represent this particular predicate and taking the sense which maximizes the cosine similarity of the predicate vector and the sense vector.

4 Results

The final system performance is shown in Table 1. The overall labeled F-score in the English in-domain data is 83.47%. When compared to our overall score in the 2014 shared task (overall labeled F-score 80.49%) a clear improvement of 3pp can be seen. This reflects the fact that predicting all arguments for a single predicate as a sequence is better than predicting them independently. The same behavior can be seen also from the methods using pure syntactic dependency parsing techniques, which have been shown to achieve the current state-of-the-art perfor-

	P	R	F	UF
Czech Open				
ID	77.53	73.20	75.30	83.03
OOD	65.11	62.35	63.70	83.10
Chinese Open				
ID	80.81	78.51	79.64	81.36

Table 2: Results for Czech and Chinese data in the open track. The Czech data is in PSD format and includes both in-domain (ID) and out-of-domain (OOD) test sets, whereas the Chinese data is in PAS format and has only in-domain test set.

	DM	PAS	PSD	Overall
Gold in-domain				
SD	88.29	**95.58**	**76.57**	86.81
DB	**93.88**	92.63	75.00	87.17
Overall-max				88.68
Gold out-of-domain				
SD	82.11	**92.92**	**75.47**	83.50
DB	**88.60**	88.93	73.43	83.65
Overall-max				85.66

Table 3: English gold track (in-domain & out-of-domain) results in terms of labeled F-score when using Stanford Dependencies (SD) and DeepBank (DB) style syntactic annotations.

mance. From out-of-domain scores we see that our system performs clearly better on in-domain data, the overall labeled F-score being 4.6pp lower when tested with out-of-domain data. Czech and Chinese open track scores are shown in Table 2.

We also provide evaluation on gold syntactic trees (gold track) using both Stanford Dependencies and DeepBank syntactic representations.[2] As can be seen from the gold track results (see Table 3) our system clearly benefits from gold-standard syntactic analyses. When comparing the performances of two syntactic representations on different formats, we can see that the optimal syntactic representation for DM format is DeepBank, whereas Stanford Dependencies fare better on PAS and PSD formats. When the best-performing syntactic representation is chosen for each format, the overall benefit on in-domain data is 5.2pp and 6.8pp on out-of-domain data compared to the open track results. Out-of-domain results improving more than in-domain results points out that better syntactic analyses help the system make more universal decisions.

[2]Unfortunately, Enju parses are not included since we could not overcome some of the problems they had in time.

The system is better at predicting relations between tokens close to each other. For example in the case of DM in-domain, relations between tokens next to each other are predicted at F-score of 92.04% while relations longer than 10 tokens at a rate of 65.11%. However, the gold syntactic analyses help predicting long relations. If we look only the relations which are ten or more tokens apart, the maximum improvement brought by gold standard syntax is 19pp for out-of-domain PAS and the minimum improvement is 6pp for in-domain DM.

5 Conclusions

Our entry in the shared task was based on an existing dependency parser, whose transition system we modified so as to essentially use the parser as a sequence classifier based on online learning and beam search. Compared to our last year's entry, the arguments of a single predicate are thus no longer predicted independently. This is accompanied by a notable gain in accuracy over the previous system which used similar features but predicted all arguments independently. From a technical point of view, basing the work on an existing parser was rather straightforward and the entire development was carried out over a period of less than two weeks.

Even though clearly better than our last year's system, the overall performance still leaves room for improvement. One possible direction would be to carry out a proper feature selection and improve the underlying machine learning algorithm of the parser to, for example, incorporate regularization. As the parser generates a large number of features optimized for syntactic parsing, it is likely that many of these are irrelevant and potentially harmful for the online-trained linear classifier.

Finally, we will evaluate the system on the Finnish PropBank data, and intend to apply it at scale to carry out SRL of the 3.2 billion token Finnish Internet Parsebank (Kanerva et al., 2014b).

Acknowledgments

This work was supported by the Emil Aaltonen Foundation and the Kone Foundation. Computational resources were provided by CSC – IT Center for Science. We would like to thank Dan Zeman for providing us the MaltParser model for Czech.

References

Bernd Bohnet and Jonas Kuhn. 2012. 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*, pages 77–87.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 570–579, Stroudsburg, PA, USA.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP'02*, pages 1–8.

Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. *SemEval 2014*, page 459.

Jan Hajič, Alena Böhmová, Eva Hajičová, and Barbora Vidová-Hladká. 2000. The Prague Dependency Treebank: A three-level annotation scenario. In *Treebanks: Building and Using Parsed Corpora*, pages 103–127.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semeckỳ, et al. 2012. Prague Czech-English Dependency Treebank 2.0.

Katri Haverinen, Veronika Laippala, Samuel Kohonen, Anna Missilä, Jenna Nyblom, Stina Ojala, Timo Viljanen, Tapio Salakoski, and Filip Ginter. 2013. Towards a dependency-based PropBank of general Finnish. In *Proceedings of the 19th Nordic Conference on Computational Linguistics (NoDaLiDa'13)*, pages 41–57.

Sam Thomson Brendan OConnor Jeffrey, Flanigan David Bamman, Jesse Dodge Swabha Swayamdipta Nathan Schneider, and Chris Dyer Noah A Smith. 2014. CMU: Arc-factored, discriminative semantic dependency parsing. *SemEval 2014*, page 176.

Jenna Kanerva and Filip Ginter. 2014. Post-hoc manipulations of vector space models with application to semantic role labeling. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC) at EACL'14*, pages 1–10.

Jenna Kanerva, Juhani Luotolahti, and Filip Ginter. 2014a. Turku: Broad-coverage semantic parsing with rich features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 678–682.

Jenna Kanerva, Matti Luotolahti, Veronika Laippala, and Filip Ginter. 2014b. Syntactic n-gram collection from a large-scale corpus of Internet Finnish. In *Proceedings of the Sixth International Conference Baltic HLT 2014*, pages 184–191.

Alexander Koller. 2014. Potsdam: Semantic dependency parsing by bidirectional graph-tree transformations and syntactic parsing. *SemEval 2014*, page 465.

Marco Kuhlmann. 2014. Linköping: Cubic-time graph parsing with a simple scoring scheme. *SemEval 2014*, page 395.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

André FT Martins and Mariana SC Almeida. 2014. Priberam: A turbo semantic parser with second order features. *SemEval 2014*, page 471.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, 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 (SemEval 2014)*, pages 63–72.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Corentin Ribeyre, Eric Villemonte de la Clergerie, and Djamé Seddah. 2014. Alpage: Transition-based semantic graph parsing with syntactic. *SemEval 2014*, page 97.

Natalie Schluter, Jakob Elming, Sigrid Klerke, Héctor Martınez Alonso, Dirk Hovy, Barbara Plank, Anders Johannsen, and Anders Søgaard. 2014. Copenhagen-Malmö: Tree approximations of semantic parsing problems. *SemEval 2014*, page 213.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(02):207–238.