# Potsdam: Semantic Dependency Parsing by Bidirectional Graph-Tree Transformations and Syntactic Parsing

**Željko Agić**
University of Potsdam
zagic@uni-potsdam.de

**Alexander Koller**
University of Potsdam
koller@ling.uni-potsdam.de

## Abstract

We present the Potsdam systems that participated in the semantic dependency parsing shared task of SemEval 2014. They are based on linguistically motivated bidirectional transformations between graphs and trees and on utilization of syntactic dependency parsing. They were entered in both the closed track and the open track of the challenge, recording a peak average labeled $F_1$ score of 78.60.

## 1 Introduction

In the semantic dependency parsing (SDP) task of SemEval 2014, the meaning of a sentence is represented in terms of binary head-argument relations between the lexical units – bi-lexical dependencies (Oepen et al., 2014). Since words can be semantic dependents of multiple other words, this framework results in graph representations of sentence meaning. For the SDP task, three such annotation layers are provided on top of the WSJ text of the Penn Treebank (PTB) (Marcus et al., 1993):

– DM: the reduction of DeepBank HPSG annotation (Flickinger et al., 2012) into bi-lexical dependencies following (Oepen and Lønning, 2006; Ivanova et al., 2012),
– PAS: the predicate-argument structures derived from the training set of the Enju HPSG parser (Miyao et al., 2004) and
– PCEDT: a subset of the tectogrammatical annotation layer from the English side of the Prague Czech-English Dependency Treebank (Cinková et al., 2009).

The three annotation schemes provide three directed graph representations for each PTB sentence, with word forms as nodes and labeled dependency relations as edges pointing from functors to arguments. The SDP-annotated PTB text is split into training (sections 00–19), development (sec. 20) and testing sets (sec. 21). This in turn makes the SDP parsing task a problem of data-driven graph parsing, in which systems are to be trained for producing dependency graph representations of sentences respecting the three underlying schemes.

While a number of theoretical and preliminary contributions to data-driven graph parsing exist (Sagae and Tsujii, 2008; Das et al., 2010; Jones et al., 2013; Chiang et al., 2013; Henderson et al., 2013), our goal here is to investigate the simplest approach that can achieve competitive performance. Our starting point is the observation that the SDP graphs are relatively tree-like. On it, we build a system for data-driven graph parsing by (1) transforming dependency graphs into dependency trees in preprocessing, (2) training and using syntactic dependency parsers over these trees and (3) transforming their output back into graphs in postprocessing. This way, we inherit the accuracy and speed of syntactic dependency parsers. The secondary benefit is insight into the structure of the semantic representations, as graph-tree transformations can make the phenomena that require non-tree-like structures more explicit.

## 2 Data and Systems

We present the basic statistics for the SDP training sets in Table 1. The graphs contain no cycles, i.e., all SDP meaning representations are directed acyclic graphs (DAGs). DM and PAS are automatically derived from HPSG annotations, while PCEDT is based on manual tectogrammatical annotation. This is reflected in more than half of the PCEDT graphs being disjoint sets of dependency trees, i.e., forests. The number of forests in DM and PAS is negligible, on the other hand. The edge





*Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 465–470,
Dublin, Ireland, August 23-24 2014.

| Feature | DM | PAS | PCEDT |
|---|---|---|---|
| Sentences | 32,389 | 32,389 | 32,389 |
| Tokens | 742,736 | 742,736 | 742,736 |
| Edge labels | 52 | 43 | 71 |
| Cyclic graphs | 0 | 0 | 0 |
| Forests | 810 | 418 | 18,527 |
| Treewidth (undirected) | 1.30 | 1.71 | 1.45 |
| Tree labels | | | |
| LOCAL | 79 | 77 | 124 |
| DFS | 79 | 81 | 133 |

Table 1: Basic statistics for the training sets.

label set of PCEDT is also substantially larger than the label sets of DM and PAS.

### 2.1 Baseline

A directed acyclic graph is a dependency tree in the sense of (Nivre, 2006) if any two nodes are connected by exactly one simple path. In other words, a DAG is a dependency tree if there are no disconnected (singleton) nodes and if there are no node reentrancies, i.e., all nodes have an indegree of 1. We calculate the average treewidth of SDP graphs by converting them to undirected graphs and applying the algorithm of (Gogate and Dechter, 2004). As we show in Table 1, the treewidth is low for all three representations. The low treewidth indicates that, even if the SDP semantic representations are graphs and not trees, these graphs are very tree-like and, as such, easily transformed into trees as there are not many edges that would require deletion. Thus, one could perform a lossy graph-to-tree conversion by (a) detecting singleton nodes and attaching them trivially and (b) detecting reentrant nodes and deleting all but one incoming edge.

The official SDP baseline system[1] (Oepen et al., 2014) is based precisely on this principle: singletons are attached to their right neighbors, only the edges to the closest predicates are kept for reentrant nodes, with a preference for leftward predicates in ties, and all remaining nodes with an indegree of 0 are attached to the root. Two dummy labels are introduced in the process: `root` for attachments to root and `null` for the remaining new attachments. The baseline is thus limited by the lossy approach to graph-to-tree reductions and the lack of linguistic motivation for these particular reduction operations. Here, we aim at introducing less lossy and more linguistically motivated reductions.

[1] `http://alt.qcri.org/semeval2014/task8/index.php?id=evaluation`

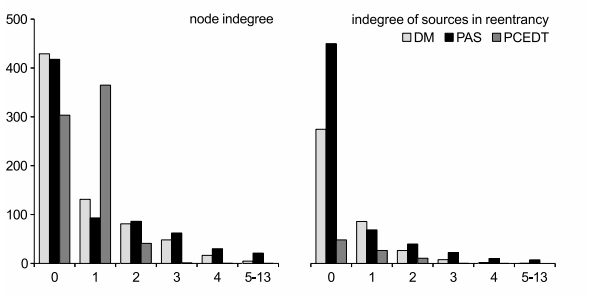


Figure 1: Distributions of node indegrees for (a) all nodes and (b) source nodes of edges participating in reentrancies.

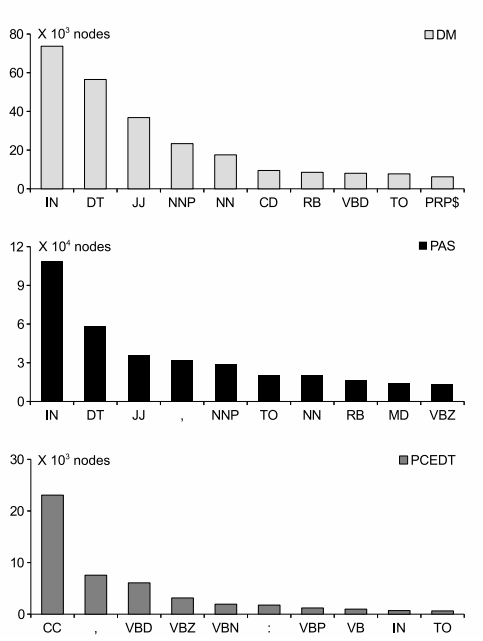


Figure 2: Distributions of parts of speech for reentrancy source nodes with zero indegree. Ten most frequent parts of speech are displayed.

### 2.2 Local Edge Flipping

Furthermore, inspecting the distribution of node indegrees in the SDP data in Figure 1, we make two important observations: (1) from its left histogram, that most of the nodes in all three annotations have an indegree of 0 or 1, and (2) from its right histogram, that most source nodes of edges causing reentrancies themselves have an indegree of 0. Figure 2 deepens this observation by providing a part-of-speech distribution of source nodes in reentrancies. It shows that the edges in DM and PAS are systematically pointed from modi-

| System | DM | PAS | PCEDT |
|---|---|---|---|
| BASELINE | 66.19 | 57.66 | 90.70 |
| LOCAL | 89.93 | 88.73 | 91.86 |
| DFS | 95.52 | 93.98 | 92.85 |

Table 2: Upper bound *LF* scores on the development set for LOCAL and DFS conversion compared to the baseline. This score indicates the quality of graph-tree transformation as no parsing is done.

| Dataset | *P* | *R* | $F_1$ |
|---|---|---|---|
| DM | 73.30 | 62.99 | 67.76 |
| PAS | 76.03 | 72.12 | 74.02 |
| PCEDT | 79.40 | 78.52 | 78.96 |

Table 3: Top node detection accuracy with CRFs on the development set for the three annotations. Precision *(P)*, recall *(R)* and the $F_1$ scores relate to marking tokens with the binary top node flag.

fiers to modifiees, while coordinating conjunctions in PCEDT introduce the coordinated nodes. We conclude that edges in reentrancies, for which the source nodes have zero indegree, could be flipped by changing places of their source and target nodes and encoding the switch in the edge labels by appending the suffix `flipped` to the existing labels.

This is the basis for our first system: LOCAL. In it, we locally flip all edges in reentrancies for which the source node has zero indegree and run the BASELINE conversion on the resulting graphs. We apply this conversion on the training data, use the converted training sets to train syntactic dependency parsers (Bohnet, 2010) and utilize the parsing models on the development and test data. The parsing outputs are converted back to graphs by simply re-flipping all the edges denoted as `flipped`.

### 2.3 Depth-first Edge Flipping

Our second system, DFS, is based on depth-first search graph traversal and edge flipping. In it, we create a undirected copy of the input graph and connect all nodes with zero indegree to the root using dummy edges. We do a depth-first traversal of this graph, starting from the root, while performing edge lookup in the original DAG. For each DFS edge traversal in the undirected copy, we check if the direction of this edge in the original DAG is identical or reversed to the traversal direction. If it is identical, we keep the existing edge. If we traverse the edge against its original direction, we reverse it. Finally, we delete the dummy edges and convert the resulting graph to a dependency tree by running the baseline, to connect the singletons to their neighbors, and to attach predicates with zero indegree and sentence-final nodes to the root.

| | DM | | PAS | | PCEDT | |
|---|---|---|---|---|---|---|
| *closed* | *LAS* | *UAS* | *LAS* | *UAS* | *LAS* | *UAS* |
| LOCAL | 79.09 | 81.35 | 81.93 | 83.79 | 81.16 | 89.60 |
| DFS | 82.02 | 83.74 | 87.06 | 87.93 | 79.94 | 88.04 |
| *open* | | | | | | |
| LOCAL | 80.86 | 82.73 | 85.16 | 86.18 | 82.04 | 90.79 |
| DFS | 84.23 | 85.77 | 88.42 | 89.26 | 80.82 | 89.02 |

Table 4: Syntactic dependency parsing accuracy of our systems before the tree-to-graph transformations, given as a set of labeled *(LAS)* and unlabeled *(UAS)* attachment scores. The scores are given for the development set.

We illustrate our graph-to-tree transformations LOCAL and DFS on a gold standard graph from the training data in Figure 3. It shows how DFS manages to preserve more edges than LOCAL by performing traversal flipping, while LOCAL flips only the edges that have source nodes with zero indegree. On the other hand, DFS performs more flipping operations than LOCAL, but as Table 1 shows, this does not result in substantial increase of the label sets.

### 2.4 Parsing and Top Node Detection

The same syntactic parser and top node detector are used in both LOCAL and DFS. Both systems ran in the closed SDP track, with no additional features for learning, and in the open track, where they used the SDP companion data, i.e., the outputs of a syntactic dependency parser (Bohnet and Nivre, 2012) and phrase-based parser (Petrov et al., 2006) as additional features. Our choice of parser was based on the high non-projectivity of the resulting trees, while parsers of (Bohnet and Nivre, 2012; Bohnet et al., 2013) could also be used, among others. We use the parser out of the box, i.e., without any parameter tuning or additional features other than what was previously listed for the open track.

Top node detection is implemented separately, by training a sequence labeling model (Lafferty et al., 2001; Kudo, 2005) on tokens and part-of-speech tags from the training sets. Its accuracy is given in Table 3. We use only the tokens and parts of speech as features for these models, and

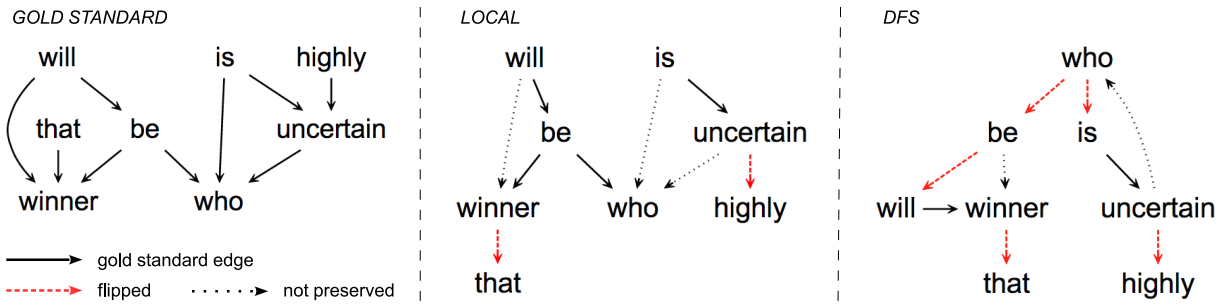


Figure 3: Illustration of graph-to-tree transformations of a gold standard graph for LOCAL and DFS. Edge labels are omitted. The sentence (PAS, #20415005): *Who that winner will be is highly uncertain.*

we design our feature set by adapting the chunking template from the CRF++ toolkit documentation.[2] We note that this model can be improved by, e.g., adding the open track companion features to the feature set, but they were not used in the experiments we present here.[3]

Our graph-to-tree conversions expand the label sets by appending the edge flip flag. The sizes of the new label sets are given in Table 1 in comparison to the original ones. The increase in size is expected to affect the parsing accuracy. The parsing accuracies on the development sets are given in Table 4. The scores correlate with the label set sizes, with a notable difference between the labeled *(LAS)* and unlabeled *(UAS)* attachment score for PCEDT. The LOCAL approach tends to outperform DFS for PCEDT, while DFS parsers also significantly outperform LOCAL for DM and PAS. The open track parsers tend to perform a little better as they make use of the additional features.

In Table 2, we measure the theoretical maximum accuracy for parsers based on our two conversions in comparison with the baseline. There, we run BASELINE, LOCAL and DFS on the development set and convert the trees back to graphs right away, i.e., without the parsing step, so as to observe the dissipation of the conversion. The scores show that LOCAL and DFS outperform BASELINE by a large margin, while the maximum accuracy for DFS is larger than the one for LOCAL, 1 point for PCEDT and around 5 points for DM and PAS. This is due to DFS performing non-local edge flipping, thus preserving more edges. The parsing scores from Table 4 and the maximum accuracy from Table 2 show that our systems are not as lossy in graph-tree conversions as the baseline, while they pay the price in the number of new labels in actual parsing and, subsequently, in the accuracy of the dependency parsers. Thus, *LAS* and *UAS* for the baseline are 1-2 points higher than the scores in Table 4 for DM and PCEDT, while our scores are 3-4 points higher for PAS.

[2] http://crfpp.googlecode.com/svn/trunk/doc/index.html

[3] The recall would increase by 15 points, amounting to a 10 point increase in $F_1$ for top node detection in DM.

| | *closed* | | *open* | |
|---|---|---|---|---|
| *dev* | *LF* | *UF* | *LF* | *UF* |
| LOCAL | 76.70 | 82.01 | 77.87 | 83.19 |
| DFS | 78.49 | 83.78 | 80.03 | 85.31 |
| *test* | | | | |
| LOCAL | 75.94 | 81.58 | 76.79 | 82.52 |
| DFS | 77.34 | 82.99 | 78.60 | 84.32 |

Table 5: Overall accuracy for our LOCAL and DFS systems, i.e., averaged labeled and unlabeled $F_1$ scores over the three annotations.

## 3 Results and Discussion

As in the official SDP scoring, we express the results in terms of labeled and unlabeled precision *(LP, UP)* and recall *(LR, UR)*, their harmonic means, the $F_1$ scores *(LF, UF)*, and sentence-level exact matches *(LM, UM)*. The official SDP scorer reports on two variants of these scores: the one taking into account the virtual edges to top nodes and the one excluding those edges. The former is less relaxed as it requires the top nodes to be predicted, and this is the only one we use in this report. We note that for our systems, the scores without the virtual edges are approximately 2 points higher for all the metrics.

The overall scores are given in Table 5. There, we provide the labeled and unlabeled $F_1$ scores on the development and test data in the closed and open track, averaged for all three annotations. The open track systems consistently score approxi-

| *closed track* | DM | | | | PAS | | | | PCEDT | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *LP* | *LR* | *LF* | *LM* | *LP* | *LR* | *LF* | *LM* | *LP* | *LR* | *LF* | *LM* |
| LOCAL | 83.39 | 72.88 | 77.78 | 4.53 | 88.18 | 74.00 | 80.47 | 2.00 | 72.25 | 67.10 | 69.58 | 6.38 |
| DFS | 79.36 | 79.34 | 79.35 | 9.05 | 88.15 | 81.60 | 84.75 | 7.72 | 69.68 | 66.25 | 67.92 | 5.86 |
| | –4.03 | +6.46 | +1.57 | +4.52 | –0.03 | +7.60 | +4.28 | +5.72 | –2.57 | –0.85 | –1.66 | –0.52 |
| | *UP* | *UR* | *UF* | *UM* | *UP* | *UR* | *UF* | *UM* | *UP* | *UR* | *UF* | *UM* |
| LOCAL | 85.47 | 74.70 | 79.72 | 5.04 | 89.70 | 75.28 | 81.86 | 2.23 | 86.36 | 80.21 | 83.17 | 19.44 |
| DFS | 81.56 | 81.54 | 81.55 | 10.31 | 89.62 | 82.96 | 86.16 | 7.86 | 83.37 | 79.27 | 81.27 | 17.51 |
| | –3.91 | +6.84 | +1.83 | +5.27 | –0.08 | +7.69 | +4.30 | +5.63 | –3.00 | –0.94 | –1.91 | –1.93 |
| *open track* | DM | | | | PAS | | | | PCEDT | | | |
| | *LP* | *LR* | *LF* | *LM* | *LP* | *LR* | *LF* | *LM* | *LP* | *LR* | *LF* | *LM* |
| LOCAL | 84.54 | 73.80 | 78.80 | 4.53 | 89.72 | 75.08 | 81.75 | 2.00 | 72.52 | 67.33 | 69.83 | 6.08 |
| DFS | 81.32 | 80.91 | 81.11 | 10.46 | 89.41 | 82.61 | 85.88 | 8.46 | 70.35 | 67.33 | 68.80 | 5.79 |
| | –3.22 | +7.11 | +2.31 | +5.93 | –0.31 | +7.53 | +4.13 | +6.46 | –2.17 | +0.00 | –1.03 | –0.29 |
| | *UP* | *UR* | *UF* | *UM* | *UP* | *UR* | *UF* | *UM* | *UP* | *UR* | *UF* | *UM* |
| LOCAL | 86.43 | 75.45 | 80.57 | 5.49 | 90.99 | 76.14 | 82.91 | 2.30 | 87.32 | 81.07 | 84.08 | 19.73 |
| DFS | 83.37 | 82.95 | 83.16 | 11.94 | 90.78 | 83.87 | 87.19 | 8.75 | 84.46 | 80.83 | 82.60 | 18.47 |
| | –3.06 | +7.50 | +2.59 | +6.45 | –0.22 | +7.73 | +4.28 | +6.45 | –2.86 | –0.24 | –1.48 | –1.26 |

Table 6: Breakdown of the scores for our LOCAL and DFS systems on the test sets. We provide labeled and unlabeled precision *(LP, UP)*, recall *(LR, UR)*, $F_1$ scores *(LF, UF)* and exact matches *(LM, UM)* for all three annotations in both the closed and the open evaluation track.

mately 1 point higher than their closed track counterparts, apparently taking advantage of the additional features available in training and testing. The DFS system is 2 points better than LOCAL in all scenarios, owing to the higher maximum coverage of the original graphs in the conversions. The large label sets amount to a difference of approximately 6 points between the labeled and unlabeled accuracies in favor of the latter attachment.

Table 6 is a breakdown of the scores in Table 5 across the three annotations and the two tracks. Here, we pair the $F_1$ scores with the corresponding precision and recall scores. We also explicitly denote the differences in scores between LOCAL and DFS. For DM and PAS, the score patterns are very similar: due to the larger label set and less regular edge flipping, DFS has a 3-4 points lower precision than LOCAL, while its recall is 6-8 points higher, amounting to the overall improvement of approximately 4 points $F_1$. In contrast, on the PCEDT data, LOCAL outperforms DFS by approximately 1.5 points. We note that the label sets for PCEDT are much larger than for DM and PAS and that the favorable reentrancies in PCEDT are much less frequent to begin with (see Table 1, Table 2 and Figure 2). At 14 points $F_1$, the discrepancy between the labeled and unlabeled scores is much higher for PCEDT than for DM and PAS, for which we observe a 1-2 point difference.

The exact match scores *(LM, UM)* favor DFS over LOCAL by approximately 5 points for DM and PAS, while LOCAL is better than DFS for PCEDT by 1-2 points. In absolute terms, the PAS scores are higher than those for DM and PAS in both our systems. This difference between the token-level and the sentence-level scores stems from the properties of our graph-tree transformations as, e.g., certain edges in undirected cycles could not be addressed by our edge inversions.

At approximately 81, 86 and 70 points $F_1$ for DM, PAS and PCEDT, in this contribution we have shown that focusing on graph-tree transformations for the utilization of a syntactic dependency parser lets us achieve good overall performance in the semantic dependency parsing task. In the future, we will further investigate what transformations are appropriate for different styles of graph-based semantic representations, and what we can learn from this both for improving SDP parser accuracy and for making linguistically motivated design choices for graph-based semantic representations. Furthermore, we will extend our system to cover inherently non-tree-like structures, such as those induced by control verbs.

**Acknowledgements** We are grateful to Stephan Oepen for all the discussions on the properties of the SDP datasets, and for providing the infrastructure for running the systems. We also thank the anonymous reviewers for their valuable insight.

## References


Bernd Bohnet and Joakim Nivre. 2012. A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. In *Proc. EMNLP-CoNLL*, pages 1455–1465.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint Morphological and Syntactic Analysis for Richly Inflected Languages. *TACL*, 1:415–428.

Bernd Bohnet. 2010. Top Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proc. COLING*, pages 89–97.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing Graphs with Hyperedge Replacement Grammars. In *Proc. ACL*, pages 924–932.

Silvie Cinková, Josef Toman, Jan Hajič, Kristýna Čermáková, Václav Klimeš, Lucie Mladová, Jana Šindlerová, Kristýna Tomšů, and Zdeněk Žabokrtský. 2009. Tectogrammatical Annotation of the Wall Street Journal. *The Prague Bulletin of Mathematical Linguistics*, 92:85–104.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic Frame-Semantic Parsing. In *Proc. NAACL*, pages 948–956.

Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank: A Dynamically Annotated Treebank of the Wall Street Journal. In *Proc. TLT*, pages 85–96.

Vibhav Gogate and Rina Dechter. 2004. A Complete Anytime Algorithm for Treewidth. In *Proc. UAI*, pages 201–208.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual Joint Parsing of Syntactic and Semantic Dependencies with a Latent Variable Model. *Computational Linguistics*, 39(4):949–998.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who Did What to Whom? A Contrastive Study of Syntacto-Semantic Dependencies. In *Proc. Linguistic Annotation Workshop*, pages 2–11.

Bevan Keeley Jones, Sharon Goldwater, and Mark Johnson. 2013. Modeling Graph Languages with Grammars Extracted via Tree Decompositions. In *Proc. FSMNLP*, pages 54–62.

Taku Kudo. 2005. CRF++: Yet another CRF toolkit. *Software available at* `http://crfpp.sourceforge.net/`.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. ICML*, pages 282–289.

Mitchell Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented Grammar Development for Acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank. In *Proc. IJCNLP*, pages 684–693.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-Based MRS Banking. In *Proc. LREC*, pages 1250–1255.

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*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proc. COLING-ACL*, pages 433–440.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-Reduce Dependency DAG Parsing. In *Proc. COLING*, pages 753–760.