Hybrid Combination of Constituency and Dependency Trees into an Ensemble Dependency Parser

Nathan David Green and Zdeněk Žabokrtský
Charles University in Prague
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Prague, Czech Republic
{green,zabokrtsky}@ufal.mff.cuni.cz

Abstract

Dependency parsing has made many advancements in recent years, in particular for English. There are a few dependency parsers that achieve comparable accuracy scores with each other but with very different types of errors. This paper examines creating a new dependency structure through ensemble learning using a hybrid of the outputs of various parsers. We combine all tree outputs into a weighted edge graph, using 4 weighting mechanisms. The weighted edge graph is the input into our ensemble system and is a hybrid of very different parsing techniques (constituent parsers, transition-based dependency parsers, and a graph-based parser). From this graph we take a maximum spanning tree. We examine the new dependency structure in terms of accuracy and errors on individual part-of-speech values.

The results indicate that using a greater number of more varied parsers will improve accuracy results. The combined ensemble system, using 5 parsers based on 3 different parsing techniques, achieves an accuracy score of 92.58%, beating all single parsers on the Wall Street Journal section 23 test set. Additionally, the ensemble system reduces the average relative error on selected POS tags by 9.82%.

1 Introduction

Dependency parsing has made many advancements in recent years. A prime reason for the quick advancement has been the CoNLL shared task competitions. These competitions gave the community a common training/testing framework along with many open source systems. These systems have, for certain languages, achieved fairly high accuracy. Many of the top systems have comparable accuracy but vary on the types of errors they make. The approaches used in the shared task vary from graph-based techniques to transition-based techniques to the conversion of constituent trees produced by state-of-the-art constituent parsers. This varied error distribution makes dependency parsing a prime area for the application of new hybrid and ensemble algorithms.

Increasing accuracy of dependency parsing often is in the realm of feature tweaking and optimization. The idea behind ensemble learning is to take the best of each parser as it currently is and allow the ensemble system to combine the outputs to form a better overall parse using prior knowledge of each individual parser. This is often done by different weighting or voting schemes.

2 Related Work

Ensemble learning (Dietterich, 2000) has been used for a variety of machine learning tasks and recently has been applied to dependency parsing in various ways and with different levels of success. (Surdeanu and Manning, 2010; Haffari et al., 2011) showed a successful combination of parse trees through a linear combination of trees with various weighting formulations. To keep their tree constraint, they applied Eisner's algorithm for reparsing (Eisner, 1996).

Parser combination with dependency trees has been examined in terms of accuracy (Sagae and Lavie, 2006; Sagae and Tsujii, 2007; Zeman and Žabokrtský, 2005). However, the various techniques have generally examined similar parsers

Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data (Hybrid2012), EACL 2012, pages 19–26,
Avignon, France, April 23 2012. ©2012 Association for Computational Linguistics

or parsers which have generated various different models. To the best of our knowledge, our experiments are the first to look at the accuracy and part of speech error distribution when combining together constituent and dependency parsers that use many different techniques. However, POS tags were used in parser combination in (Hall et al., 2007) for combining a set of Malt Parser models with success.

Other methods of parser combinations have shown to be successful such as using one parser to generate features for another parser. This was shown in (Nivre and McDonald, 2008), in which Malt Parser was used as a feature to MST Parser. The result was a successful combination of a transition-based and graph-based parser, but did not address adding other types of parsers into the framework.

3 Methodology

The following sections describe the process flow, choice of parsers, and datasets needed for others to recreate the results listed in this paper. Although we describe the specific parsers and datasets used in this paper, this process flow should work for any number of hybrid combinations of parsers and datasets.

3.1 Process Flow

To generate a single ensemble parse tree, our system takes N parse trees as input. The inputs are from a variety of parsers as described in 3.2. All edges in these parse trees are combined into a graph structure. This graph structure accepts weighted edges. So if more than one parse tree contains the same tree edge, the graph is weighted appropriately according to a chosen weighting algorithm. The weighting algorithms used in our experiments are described in 3.5.

Once the system has a weighted graph, it then uses an algorithm to find a corresponding tree structure so there are no cycles. In this set of experiments, we constructed a tree by finding the maximum spanning tree using ChuLiu/Edmonds' algorithm, which is a standard choice for MST tasks. Figure 1 graphically shows the decisions one needs to make in this framework to create an ensemble parse.

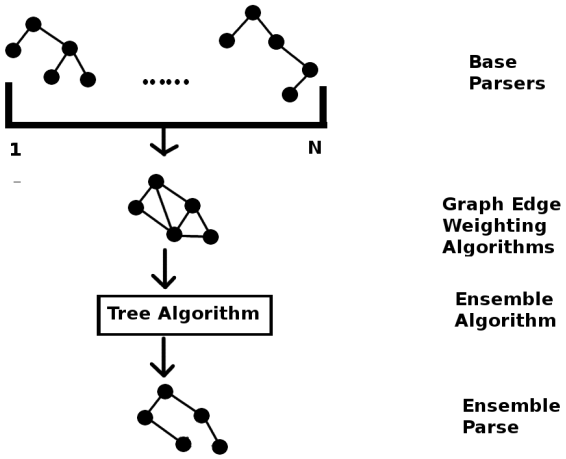

Figure 1: General flow to create an ensemble parse tree.

3.2 Parsers

To get a complete representation of parsers in our ensemble learning framework we use 5 of the most commonly used parsers. They range from graph-based approaches to transition-based approaches to constituent parsers. Constituency output is converted to dependency structures using a converter (Johansson and Nugues, 2007). All parsers are integrated into the Treex framework (Žabokrtský et al., 2008; Popel et al., 2011) using the publicly released parsers from the respective authors but with Perl wrappers to allow them to work on a common tree structure.

- **Graph-Based:** A dependency tree is a special case of a weighted edge graph that spawns from an artificial root and is acyclic. Because of this we can look at a large history of work in graph theory to address finding the best spanning tree for each dependency graph. In this paper we use MST Parser (McDonald et al., 2005) as an input to our ensemble parser.

- **Transition-Based:** Transition-based parsing creates a dependency structure that is parameterized over the transitions used to create a dependency tree. This is closely related to shift-reduce constituency parsing algorithms. The benefit of transition-based parsing is the use of greedy algorithms which have a linear time complexity. However, due to the greedy algorithms, longer arc parses can cause error propagation across each transition (Kübler et al., 2009). We make use

of Malt Parser (Nivre et al., 2007b), which in the shared tasks was often tied with the best performing systems. Additionally we use Zpar (Zhang and Clark, 2011) which is based on Malt Parser but with a different set of non-local features.

- **Constituent Transformation** While not a true dependency parser, one technique often applied is to take a state-of-the-art constituent parser and transform its phrase based output into dependency relations. This has been shown to also be state-of-the-art in accuracy for dependency parsing in English. In this paper we transformed the constituency structure into dependencies using the Penn Converter conversion tool (Johansson and Nugues, 2007). A version of this converter was used in the CoNLL shared task to create dependency treebanks as well. For the following ensemble experiments we make use of both (Charniak and Johnson, 2005) and Stanford's (Klein and Manning, 2003) constituent parsers.

In addition to these 5 parsers, we also report the accuracy of an Oracle Parser. This parser is simply the best possible parse of all the edges of the combined dependency trees. If the reference, gold standard, tree has an edge that any of the 5 parsers contain, we include that edge in the Oracle parse. Initially all nodes of the tree are attached to an artificial root in order to maintain connectedness. Since only edges that exist in a reference tree are added, the Oracle Parser maintains the acyclic constraint. This can be viewed as the maximum accuracy that a hybrid approach could achieve with this set of parsers and with the given data sets.

3.3 Datasets

Much of the current progress in dependency parsing has been a result of the availability of common data sets in a variety of languages, made available through the CoNLL shared task (Nivre et al., 2007a). This data is in 13 languages and 7 language families. Later shared tasks also released data in other genres to allow for domain adaptation. The availability of standard competition, gold level, data has been an important factor in dependency based research.

For this study we use the English CoNLL data. This data comes from the Wall Street Journal (WSJ) section of the Penn treebank (Marcus et al., 1993). All parsers are trained on sections 02-21 of the WSJ except for the Stanford parser which uses sections 01-21. Charniak, Stanford and Zpar use pre-trained models *ec50spfinal*, *wsjPCFG.ser.gz*, *english.tar.gz* respectively. For testing we use section 23 of the WSJ for comparability reasons with other papers. This test data contains 56,684 tokens. For tuning we use section 22. This data is used for determining some of the weighting features.

3.4 Evaluation

As an artifact of the CoNLL shared tasks competition, two standard metrics for comparing dependency parsing systems emerged. *L*abeled *a*ttachment *s*core (LAS) and *u*nlabeled *a*ttachment *s*core (UAS). UAS studies the structure of a dependency tree and assesses whether the output has the correct head and dependency arcs. In addition to the structure score in UAS, LAS also measures the accuracy of the dependency labels on each arc. A third, but less common metric, is used to judge the percentage of sentences that are completely correct in regards to their LAS score. For this paper since we are primarily concerned with the merging of tree structures we only evaluate UAS (Buchholz and Marsi, 2006).

3.5 Weighting

Currently we are applying four weighting algorithms to the graph structure. First we give each parser the same uniform weight. Second we examine weighting each parser output by the UAS score of the individual parser taken from our tuning data. Third we use plural voting weights (De Pauw et al., 2006) based on parser ranks from our tuning data. Due to the success of Plural voting, we try to exaggerate the differences in the parsers by using UAS^{10} weighting. All four of these are simple weighting techniques but even in their simplicity we can see the benefit of this type of combination in an ensemble parser.

- **Uniform Weights**: an edge in the graph gets incremented +1 weight for each matching edge in each parser. If an edge occurs in 4 parsers, the weight is 4.

- **UAS Weighted**: Each edge in the graph gets

incremented by the value of it's parsers individual accuracy. So in the UAS results in Table 1 an edge in Charniak's tree gets .92 added while MST gets .86 added to every edge they share with the resulting graph. This weighting should allow us to add poor parsers with very little harm to the overall score.

- **Plural Voting Weights**: In Plural Voting the parsers are rated according to their rank in our tuning data and each gets a "vote" based on their quality. With N parsers the best parser gets N votes while the last place parser gets 1 vote. In this paper, Charniak received 5 votes, Stanford received 4 votes, MST Parser received 3 votes, Malt Parser received 2 votes, and Zpar received 1 vote. Votes in this case are added to each edge as a weight.

- **UAS**^{10}: For this weighting scheme we took each UAS value to the 10th power. This gave us the desired affect of making the differences in accuracy more apparent and giving more distance from the best to worse parser. This exponent was empirically selected from results with our tuning data set.

4 Results

Table 1 contains the results of different parser combinations of the 5 parsers and Table 2 shows the baseline scores of the respective individual parsers. The results indicate that using two parsers will result in an "average" score, and no combination of 2 parsers gave an improvement over the individual parsers, these were left out of the table. Ensemble learning seems to start to have a benefit when using 3 or more parsers with a few combinations having a better UAS score than any of the baseline parsers, these cases are in bold throughout the table. When we add a 4th parser to the mix almost all configurations lead to an improved score when the edges are not weighted uniformly. The only case in which this does not occur is when Stanford's Parser is not used.

Uniform voting gives us an improved score in a few of the model combinations but in most cases does not produce an output that beats the best individual system. UAS weighting is not the best overall but it does give improved performance in the majority of model combinations. Problematically UAS weighted trees do not give an improved accuracy when all 5 parsers are used. Given the slight differences in UAS scores of the baseline models in Table 2 this is not surprising as the best graph edge can be outvoted as the number of N parsers increases. The slight differences in weight do not seem to change the MST parse dramatically when all 5 parsers are used over Uniform weighting. Based on the UAS scores learned in our tuning data set, we next looked to amplify the weight differences using Plural Voting. For the majority of model combinations in Plural voting we achieve improved results over the individual systems. When all 5 parsers are used together with Plural Voting, the ensemble parser improves over the highest individual parser's UAS score. With the success of Plural voting we looked to amplify the UAS score differences in a more systematic way. We looked at using UAS^x where x was found experimentally in our tuning data. UAS^{10} matched Plural voting in the amount of system combinations that improved over their individual components. The top overall score is when we use UAS^{10} weighting with all parsers. For parser combinations that do not feature Charniak's parser, we also find an increase in overall accuracy score compared to each individual parser, although never beating Charniak's individual score.

To see the maximum accuracy a hybrid combination can achieve we include an Oracle Ensemble Parser in Table 1. The Oracle Parser takes the edges from all dependency trees and only adds each edge to the Oracle Tree if the corresponding edge is in the reference tree. This gives us a ceiling on what ensemble learning can achieve. As we can see in Table 1, the ceiling of ensemble learning is 97.41% accuracy. Because of this high value with only 5 parsers, ensemble learning and other hybrid approaches should be a very prosperous area for dependency parsing research.

In (Kübler et al., 2009) the authors confirm that two parsers, MST Parser and Malt Parser, give similar accuracy results but with very different errors. MST parser, a maximum spanning tree graph-based algorithm, has evenly distributed errors while Malt Parser, a transition based parser, has errors on mainly longer sentences. This re-

System	Uniform Weighting	UAS Weighted	Plural Voting	UAS^{10} Weighted	Oracle UAS
Charniak-Stanford-Mst	91.86	**92.27**	**92.28**	**92.25**	96.48
Charniak-Stanford-Malt	91.77	**92.28**	**92.3**	92.08	96.49
Charniak-Stanford-Zpar	91.22	91.99	92.02	92.08	95.94
Charniak-Mst-Malt	88.80	89.55	90.77	92.08	96.3
Charniak-Mst-Zpar	90.44	91.59	92.08	92.08	96.16
Charniak-Malt-Zpar	88.61	91.3	92.08	92.08	96.21
Stanford-Mst-Malt	87.84	**88.28**	**88.26**	**88.28**	95.62
Stanford-Mst-Zpar	**89.12**	**89.88**	**88.84**	**89.91**	95.57
Stanford-Malt-Zpar	**88.61**	**89.57**	87.88	87.88	95.47
Mst-Malt-Zpar	**86.99**	**87.34**	**86.82**	86.49	93.79
Charniak-Stanford-Mst-Malt	90.45	**92.09**	**92.34**	**92.56**	97.09
Charniak-Stanford-Mst-Zpar	91.57	**92.24**	**92.27**	**92.26**	96.97
Charniak-Stanford-Malt-Zpar	91.31	**92.14**	**92.4**	**92.42**	97.03
Charniak-Mst-Malt-Zpar	89.60	89.48	91.71	92.08	96.79
Stanford-Mst-Malt-Zpar	**88.76**	**88.45**	**88.95**	**88.44**	96.36
All	91.43	91.77	**92.44**	**92.58**	97.41

Table 1: Results of the maximum spanning tree algorithm on a combined edge graph. Scores are in **bold** when the ensemble system increased the UAS score over all individual systems.

Parser	UAS
Charniak	92.08
Stanford	87.88
MST	86.49
Malt	84.51
Zpar	76.06

Table 2: Our baseline parsers and corresponding UAS used in our ensemble experiments

sult comes from the approaches themselves. MST parser is globally trained so the best mean solution should be found. This is why errors on the longer sentences are about the same as the shorter sentences. Malt Parser on the other hand uses a greedy algorithm with a classifier that chooses a particular transition at each vertex. This leads to the possibility of the propagation of errors further in a sentence. Along with this line of research, we look at the error distribution for all 5 parsers along with our best ensemble parser configuration. Much like the previous work, we expect different types of errors, given that our parsers are from 3 different parsing techniques. To examine if the ensemble parser is substantially changing the parse tree or is just taking the best parse tree and substituting a few edges, we examine the part of speech accuracies and relative error reduction in Table 3.

As we can see the range of POS errors varies dramatically depending on which parser we examine. For instance for *CC*, Charniak has 83.54% accuracy while MST has only 71.16% accuracy. The performance for certain POS tags is almost universally low such as the left parenthesis *(*. Given the large difference in POS errors, weighting an ensemble system by POS would seem like a logical choice in future work. As we can see in Figure 2, the varying POS accuracies indicate that the parsing techniques we have incorporated into our ensemble parser, are significantly different. In almost every case in Table 3, our ensemble parser achieves the best accuracy for each POS, while reducing the average relative error rate by 9.82%.

The current weighting systems do not simply default to the best parser or to an average of all errors. In the majority of cases our ensemble parser obtains the top accuracy. The ability of the ensemble system to use maximum spanning tree on a graph allows the ensemble parser to connect nodes which might have been unconnected in a subset of the parsers for an overall gain, which is preferable to techniques which only select the best model for a particular tree. In all cases, our ensemble parser is never the worst parser. In

POS	Charniak	Stanford	MST	Malt	Zpar	Best Ensemble	Relative Error Reduction
CC	83.54	74.73	71.16	65.84	20.39	**84.63**	6.62
NNP	94.59	92.16	88.04	87.17	73.67	**95.02**	7.95
VBN	91.72	89.81	90.35	89.17	88.26	**93.81**	25.24
CD	94.91	92.67	85.19	84.46	82.64	**94.96**	0.98
RP	96.15	95.05	97.25	95.60	94.51	**97.80**	42.86
JJ	95.41	92.99	94.47	93.90	89.45	**95.85**	9.59
PRP	97.82	96.21	96.68	95.64	95.45	**98.39**	26.15
TO	94.52	89.44	91.29	90.73	88.63	94.35	-3.10
WRB	63.91	60.90	68.42	73.68	4.51	63.91	0.00
RB	86.26	79.88	81.49	81.44	80.61	**87.19**	6.77
WDT	97.14	95.36	96.43	95.00	9.29	**97.50**	12.59
VBZ	91.97	87.35	83.86	80.78	57.91	**92.46**	6.10
(	73.61	75.00	54.17	58.33	15.28	73.61	0.00
POS	98.18	96.54	98.54	98.72	0.18	**98.36**	9.89
VB	93.04	88.48	91.33	90.95	84.37	**94.24**	17.24
MD	89.55	82.02	83.05	78.77	51.54	**89.90**	3.35
NNS	93.10	89.51	90.68	88.65	78.93	**93.67**	8.26
NN	93.62	90.29	88.45	86.98	83.84	**94.00**	5.96
VBD	93.25	87.20	86.27	82.73	64.32	**93.52**	4.00
DT	97.61	96.47	97.30	97.01	92.19	**97.97**	15.06
RBS	90.00	76.67	93.33	93.33	86.67	90.00	0.00
IN	87.80	78.66	83.45	80.78	73.08	87.48	-2.66
)	70.83	77.78	96.46	55.56	12.50	**72.22**	4.77
VBG	85.19	82.13	82.74	82.25	81.27	**89.35**	28.09
Average							**9.82**

Table 3: POS accuracies for each of our systems that are used in the ensemble system. We use these accuracies to obtain the POS error distribution for our best ensemble system, which is the combination of all parsers using UAS^{10} weighting. Relative error reduction is calculated between our best ensemble system against the Charniak Parser which had the best individual scores.

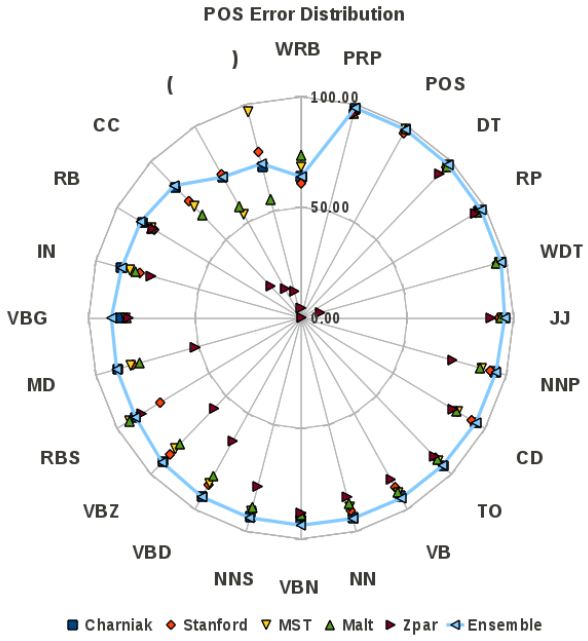

Figure 2: POS errors of all 5 parsers and the best ensemble system

cases where the POS is less frequent, our ensemble parser appears to average out the error distribution.

5 Conclusion

We have shown the benefits of using a maximum spanning tree algorithm in ensemble learning for dependency parsing, especially for the hybrid combination of constituent parsers with other dependency parsing techniques. This ensemble method shows improvements over the current state of the art for each individual parser. We also show a theoretical maximum oracle parser which indicates that much more work in this field can take place to improve dependency parsing accuracy toward the oracle score of 97.41%.

We demonstrated that using parsers of different techniques, especially including transformed constituent parsers, can lead to the best accuracy within this ensemble framework. The improvements in accuracy are not simply due to a few edge changes but can be seen to improve the accuracy of the majority of POS tags over all individual systems.

While we have only shown this for English, we expect the results to be similar for other languages since our methodology is language independent. Future work will contain different weighting mechanisms as well as application to other languages which are included in CoNLL data sets.

6 Acknowledgments

This research has received funding from the European Commission's 7th Framework Program (FP7) under grant agreement n° 238405 (CLARA)

References

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Guy De Pauw, Gilles-Maurice de Schryver, and Peter Wagacha. 2006. Data-driven part-of-speech tagging of kiswahili. In Petr Sojka, Ivan Kopecek, and Karel Pala, editors, *Text, Speech and Dialogue*, volume 4188 of *Lecture Notes in Computer Science*, pages 197–204. Springer Berlin / Heidelberg.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, pages 1–15, London, UK. Springer-Verlag.

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, Copenhagen, August.

Gholamreza Haffari, Marzieh Razavi, and Anoop Sarkar. 2011. An ensemble model that combines syntactic and semantic clustering for discriminative dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 710–714, Portland, Oregon, USA, June. Association for Computational Linguistics.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for

English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

S. Kübler, R. McDonald, and J. Nivre. 2009. *Dependency parsing*. Synthesis lectures on human language technologies. Morgan & Claypool, US.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the Penn Treebank. *Comput. Linguist.*, 19:313–330, June.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gulsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Martin Popel, David Mareček, Nathan Green, and Zdeněk Žabokrtský. 2011. Influence of parser choice on dependency-based mt. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 433–439, Edinburgh, Scotland, July. Association for Computational Linguistics.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA, June. Association for Computational Linguistics.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050, Prague, Czech Republic, June. Association for Computational Linguistics.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 649–652, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas. 2008. TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer. In *Proceedings of the 3rd Workshop on Statistical Machine Translation, ACL*, pages 167–170.

Daniel Zeman and Zdeněk Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *In: Proceedings of the 9th International Workshop on Parsing Technologies*.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.