ITAT 2016 Proceedings, CEUR Workshop Proceedings Vol. 1649, pp. 42–47
http://ceur-ws.org/Vol-1649, Series ISSN 1613-0073, © 2016 T. Jelínek

ITAT

Partial accuracy rates and agreements of parsers: two experiments with ensemble parsing of Czech

Tomáš Jelínek

Charles University, Prague, Czech Republic
Tomas.Jelinek@ff.cuni.cz

Abstract: We present two experiments with ensemble parsing, in which we obtain a 1.4% improvement of UAS compared to the best parser. We use five parsers: MateParser, TurboParser, Parsito, MaltParser a MSTParser, and the data of the analytical layer of Prague Dependency Treebank (1.5 million tokens). We split training data into 10 data-splits and run a 10-fold cross-validation scheme with each of the five parsers. In this way, we obtain large parsed data to experiment with. In one experiment, we calculate partial accuracy rates of each parser according to a list of parameters, which we then use as weights in a combination of parsers using an algorithm for finding the maximum spanning tree. In the other experiment, we calculate success rates for agreements of parsers (e.g. Mate+MST vs. Turbo+Malt), and use these rates in another combination of parsers. Both experiments achieve an UAS above 90.0% (1.4% higher than TurboParser), the experiment with accuracy rates achieves better LAS.

1 Introduction

For some tasks in NLP (such as corpus annotation, creation of gold standard using human corrected parser output etc.), the accuracy of dependency parsing is far more important than parsing speed. For such cases, ensemble parsing (the combination of several parsers) may do the best job. In this paper, we present two experiments with ensemble parsing, in which we obtain a 1.4% improvement of UAS compared to the best parser. We use five parsers and the data of the analytical layer of Prague Dependency Treebank. We run a 10-fold cross-validation scheme over the training data with each of the five parsers. In this way, we obtain large parsed data to experiment with. In one experiment, we calculate partial accuracy rates of each parser (e. g. the proportion of correct attachments of a token with a given POS to another token), which we then use as weights in a combination of parsers. In another experiment, we calculate a success rate for agreements of parsers (e. g. Mate+MST vs. Turbo+Malt), and use these rates in another combination of parsers.
We focus only on Czech, as our main goal is to create a well parsed Czech treebank, but we plan to test our approach on other languages, in subsection 6.3 we enumerate the steps necessary to reproduce our experiments on other languages.
Similar experiments with ensemble parsing have been performed, e. g. [3] and [2] for the first experiment and [10] for the second one.

The work has been supported by the grant 16-07473S (Between lexicon and grammar) of the Grant Agency of the Czech Republic.

2 Parsers and data

In our experiments with ensemble parsing, we use five dependency parsers: TurboParser [6], a dependency parser included in Mate-tools [1] (MateParser), Parsito [9], MaltParser [8] and MSTParser [7]. The experiments are based on the data from the analytical layer of Prague Dependency Treebank[4] (PDT: 1.5 million tokens, 80.000 sentences). PDT data are split into training data (1.170.000 tokens), development test data (dtest, 159.000 tokens) and evaluation test data (etest, 174.000 tokens). We performed morphological tagging of the data using the Featurama tagger[1] with a precision of 95.2%. One of the parsers, Mate-tools, does its own tagging, with a slightly lower precision of 94.1%.
In the two following sub-sections, we describe two steps we take before the training of the parsers and parsing in order to improve parsing accuracy. They are not directly related to the subject of this paper, but they influence the results of the experiments.

2.1 Text simplification tool

In previous experiments with parsing, we found out that parsing accuracy can be significantly increased by reducing the variability of the text.
In the process of training, the parsers create a language model based on the training data. Because of phenomena like valency the parsers cannot rely on morphological tags only, they need to consider lemmas (and occasionally forms) of the tokens. But the data are sparse, in PDT 45% of lemmas occur only once and many more Czech lemmas are completely out-of-vocabulary. Consequently, the model formed by the parser is incomplete which limits the quality of parsing new text.
We have devised (see [5]) a partial solution to this problem: a text simplification tool. In many syntactic constructions, the choice of any lemma inside a group of words yields the same dependency tree: *president Clinton / Bush / Obama declared*. We identify members of about fifty

[1] See http://sourceforge.net/projects/featurama/.

such groups of words with identical syntactic properties and replace them with one representative member for each group. The text loses information (kept in a backup file), but the reduced variability facilitates parsing. Both training and new data are simplified. The variability of lemmas in text is reduced by approx. 20%, resulting in an increase of parsing accuracy of 0.5–1.5% (some parsers, e. g. Malt, benefit more from text simplification than others, e. g. MST). Mate-tools lemmatizes and tags the text itself, and therefore it could not use our simplification: only a limited simplification of the raw data (based mostly on word forms) is performed.

2.2 MWE identification and replacement

We use a list of multi-word expressions with suitable syntactic properties and replace them in the text (both training data and new text to be parsed) by one proxy item. This replacement can be only if either there cannot be any tokens dependent on any member of the MWE, or it is known to which token of the MWE each dependent token has to be attached. Our list of MWEs includes compound words, e. g. compound prepositions such as *v souvislosti s* 'relating to', phrasemes/idioms (*ležet ladem* 'lie fallow') and multi-word named entities (*Kolín nad Rýnem* 'Cologne upon the Rhein').

2.3 Parsing the training data

In order to obtain detailed information on the behavior of the parsers, we parse all the training data (1.2 million tokens) using a 10-fold cross-validation scenario (the training data are split into 10 parts, we use 90% as training data and 10% as test data in 10 iterations) with each of the five parsers. Using these data, we test two approaches to ensemble parsing.

2.4 Parsing the test data

All five parsers were also trained on the whole training data (1.2 M tokens) and used to parse PDT dtest and etest data (approx. 150.000 tokens each). The output of the parsers was then merged in one file to allow experiments with ensemble parsing. Table 1 shows the accuracy of the parsers on PDT etest data. Four accuracy measures are shown: UAS and LAS (unlabeled and labeled attachment score for single tokens), SENT_U and SENT_L (unlabeled and labeled attachment score for the whole sentences). TurboParser achieved the best UAS score (88.63%), but performed only slightly better than MateParser, which has all four scores very high (TurboParser has comparatively poor labeled scores).

3 Analysis of merged parsed data

The results of the parsing by the five parsers of all the data (train data, dev. test, eval. test) are merged in three files.

Table 1: Accuracy of the parsers (etest)

	UAS	LAS	SENT_U	SENT_L
Mate	88.58	83.09	45.87	33.77
Turbo	88.63	82.37	44.86	28.75
Malt	86.74	81.32	42.40	32.68
Parsito	86.71	81.42	41.81	32.65
MST	86.41	79.30	38.93	24.64

We use merged train data to gather information on the behaviour of parsers for the purpose of the ensemble parsing experiments, dev. test is used for fine-tuning both approaches, eval. test is used for final testing.
In this section, we provide a brief analysis of the parsed data based on the dev. test. We count how frequently the parsers agree among one another and what the accuracy corresponding to the occurrences is when a given number of parsers agree. We calculate a hypothetical floor and ceiling for the accuracy rates (UAS, LAS etc.) of any ensemble parsing experiment using these data. We detect and count potential cycles in the data.

3.1 Agreements and disagreements of parsers

In the dev. test data, we calculate how often any given number of parsers agree on a dependency relation (unlabeled scores) or on a dependency relation and a dependency label (labeled scores), then we calculate the accuracy rate of the dependency relation chosen by the highest number of parsers. For example, we find 8330 tokens for which any three parsers agree on one dependency relation and two other parsers agree on another one ("3+2" in Table 2), and the proportion of correct tokens chosen by three parsers in these 8330 tokens is 56.95%.
Table 2 and 3 present these statistics for unlabeled and labeled relations, respectively. The first column indicates the size (number) of agreeing groups of parsers ("5" means all parsers agree, "2+2+1" means two parsers agree on one dep. relation, other two parsers agree on another one, one parser has chosen a third possible dependency relation). The second column shows the number of such occurrences in dev. test data. The third column shows the accuracy, i. e. the portion of correct dep. relations chosen by the highest number of parsers; for "2+2+1" and "1+1+1+1+1", the number expresses the accuracy of a random choice (number of occurrences when at least one of the two pairs or five individual parsers is correct divided by two or five, respectively).

For 88.68% of the tokens, four or five parsers agree on an unlabeled dependency relation, with an unlabeled accuracy rate of 94.99%.

For labeled agreements, the parsers disagree more frequently and the accuracy is lower, but for the majority of tokens, 83.24%, four or five parsers agree, with a labeled accuracy of 92.59%.

Table 2: Unlabeled agreements of parsers (dtest)

Agree	Occurrences	Accuracy
5	123751	97.32
4+1	17215	78.32
3+2	8330	56.95
3+1+1	4515	58.51
2+2+1	2830	35.38
2+1+1+1	2003	35.90
1+1+1+1+1	318	14.33

Table 3: Labeled agreements of parsers (dtest)

Agree	Occurrences	Accuracy
5	111083	95.90
4+1	21237	75.31
3+2	10221	54.59
3+1+1	7058	54.61
2+2+1	4117	33.57
2+1+1+1	4133	32.71
1+1+1+1+1	1113	11.16

3.2 Floor and ceiling

We calculate a hypothetical floor and ceiling for any ensemble parsing experiment using these data: the floor is the worst possible outcome of any experiment (every token, for which at least one parser has an incorrect dep. relation (or label) is considered incorrect), the ceiling is the best possible outcome (if at least one parser has found the correct dep. relation, the token is counted as correct).
We calculate also the floor and ceiling for a simple combination of parsers, in which the dependency relation (or a labeled dep. relation) for which the most parsers agree is always taken. Only if all parsers disagree or two pairs of parsers disagree, the incorrect attachments are counted for the floor of the combination and correct attachments (if any) are counted for the ceiling of the combination.
In neither case, the cycles formed are counted or resolved, therefore the numbers do not reflect accurately the possibilities of a real ensemble parsing experiment.
Table 4 shows the accuracy rates for the floor and ceiling of any experiment and of a simple combination. The difference in accuracy measures between the floor and the ceiling of the simple combination is small, because a decision has to be made only for approx. 2% of the tokens (when all five parsers disagree or two pairs of parsers and one single parser each choose a different dep. relation).

Table 4: Floor and ceiling for ensemble parsing (dtest)

	UAS	LAS	SENT_U	SENT_L
Floor any	75.76	67.02	25.95	16.84
Floor comb.	89.34	83.86	46.29	33.46
Ceiling comb.	90.75	85.99	48.74	36.12
Ceiling any	95.72	92.55	69.03	55.27

3.3 Potential cycles

We calculate also the number of sentences, where a combination of the results of the five parsers may form a cycle. If any unlabeled dependency relation proposed by any parser can be chosen, the cycles can form in 46.35% of the sentences. For the simple combination described above, a cycle can form in 8.76% of sentences.

4 Ensemble parsing using partial accuracy rates

Our first approach to ensemble parsing is based on the observation (experimentally confirmed) that each parser tends to make consistently the same types of mistakes when using similar training and testing data. Using parsed training data, we determine the strengths and weaknesses of each parser and use them as additional input when combining the parses of new sentences.

4.1 Partial accuracy rates

Based on the parsed training data, we calculate partial accuracy rates for each parser, comparing parsed data with the gold standard. These rates are calculated as the ratio of correct attachments (and labels, in case of labeled rates) of tokens with a given morphosyntactic parameter (e. g. POS) in the total number of such tokens, partial accuracy rates have values between 0 and 1. For example, an accuracy rate 0.92 calculated for the MateParser for the unlabeled parameter POS2POS with the value "NV" means that among all dependency relations with nouns as dependent tokens and verbs as governing tokens, 92% are correct. Twelve parameters are calculated using more or less fine-grained morphosyntactic and syntactic parameters: overall accuracy of the parsers, POS of the dependent token and POS of the governing token, the distance between the dependent and the governing tokens (11 intervals: distance 0/root, 1, 2–3, 4–6, 7–10, 11 and more, dependent to the left or to the right), POS and more detailed morphological properties of the dependent token (subtype of POS and case). There are approx. 1400 values altogether for each parser (7000 values in the table of partial accuracy rates).
Table 5 presents a fraction of the table of partial accuracy rates: two values of the unlabeled parameter POS2POS calculated for all five parsers. The value "NA" indicates nouns attached to adjectives, as in *plný ryb* 'full of fish', "NV" denotes nouns attached to verbs, e. g. *chytil rybu* 'he caught a fish'.

4.2 Ensemble parsing using the MST algorithm

These partial accuracy rates (of a chosen parameter or combination of parameters) are used as weights of edges in ensemble parsing, where all five parses of a sentence

Table 5: Example of partial accuracy rates

Parser	parameter	un/lab	value	e. rate
Malt	POS2POS	UAS	NA	0.769
Mate	POS2POS	UAS	NA	0.796
MST	POS2POS	UAS	NA	0.757
Parsito	POS2POS	UAS	NA	0.741
Turbo	POS2POS	UAS	NA	0.810
Malt	POS2POS	UAS	NV	0.898
Mate	POS2POS	UAS	NV	0.921
MST	POS2POS	UAS	NV	0.894
Parsito	POS2POS	UAS	NV	0.903
Turbo	POS2POS	UAS	NV	0.909

are merged into one oriented graph. If some parsers agree on an edge (dependency relation), the sum of the accuracy rates of the parsers is used. An exponent can be also included in the calculation of weights: it raises the accuracy rate to the power of the chosen number (e. g. 0.741^6), increasing the differences between good and bad error rates, as suggested in [3].

We use Chu-Liu-Edmonds' algorithm to find the maximum spanning tree in the graph (see [2], p. 526), determining the best outcome of the combination of dependency parses of any sentence according to the chosen parameter. If parsers agree on a dependency relation, but disagree on a dependency label, weights (labeled, even if unlabeled parameter is chosen for edges) are also used to determine the best label.

Using PDT dtest data, we run a series of experiments with various parameters and combinations of parameters to determine the best parameter and exponent to use for the calculation of weights.

The results vary between the baseline (MateParser) and a 1.4/1.7% increase in UAS/LAS. Table 6 shows six examples of ensemble parsing using PDT dtest data, with various parameters (UAS/LAS and exponent for the best results with the given parameter are chosen). The first column indicates the parameter used, the second one indicates whether labeled or unlabeled attachments were used to calculate error rates, the third column presents the exponent. LAS, UAS, SENT_U and SENT_L scores are shown. The accuracy scores of MateParser are included in the table as baseline.

"ALL" parameter reflects the overall accuracy of each parser (UAS or LAS score). "2POS" parameter is based on POS of the governing token. "POS" parameter is based on POS of the dependent token. "POS2POS" combines both. "POSCASE" uses POS of the dependent token and its case. "DIST" parameter expresses the distance between the governing and dependent tokens (see subsection 4.1). For each parameter (and some of their combinations), 18 tests of ensemble parsing were run, with labeled and unlabeled accuracy rates and exponents of 1 to 9 (in our tests, higher exponents than 9 never led to an increase in accuracy).

Table 6: Tests of parameters for ensemble parsing (dtest)

Parameter	un/lab	exp.	UAS	LAS	SENT_U	SENT_L
MateParser			88.62	83.11	45.91	33.79
2POS	LAS	1	88.82	83.62	41.70	31.47
DIST	UAS	2	89.67	84.09	46.01	32.98
ALL	LAS	1	89.74	84.43	47.20	34.17
POS	LAS	4	89.82	84.53	47.22	34.20
POSCASE	LAS	2	90.02	84.76	47.51	34.54
POS2POS	UAS	6	90.07	84.83	47.61	34.78

The best results were obtained with the parameter POS2POS, unlabeled, with the exponent 6. For some combinations of two or more parameters (for example, POS:LAS+POSSUBPOS:LAS, with exp. 4 achieves an accuracy of 89.83 / 84.55 / 47.24 / 34.26), we did get better than average results, but no such combination has achieved better accuracy in all categories than the POS2POS parameter.

5 Ensemble parsing using agreements of parsers

Our second approach to ensemble parsing stems from the observation of the interaction of parsers. Using the parsed training data, we calculate how reliable parsers are in the task of assigning dependency relations to tokens, when they agree or disagree with other parsers. We sort pairs and triples of parsers by their accuracy and use this piece of information to choose the dependency relation determined by the most reliable combination of parsers. A similar (simpler) approach was proposed in [10].

5.1 Accuracy rates of agreements of parsers

We start with a file containing the training data parsed by all five parsers, the same way as in the case of our first approach with error rates. From these data, we calculate a reliability rate (accuracy) of "agreements" of parsers, i.e. of instances when two or more parsers agree on a prediction of a dependency relation for a token and some other parsers disagree.

We count the number of occurrences when a group of parsers (or just one single parser) chooses a dependency relation for a token and another group agree on another (or the others disagree), and the number of occurrences when such a choice is correct. For example, there are approx. 10.000 cases when Mate, Turbo and MST agree on a dependency relation for a token and Malt and Parsito agree on another one. In 62.8% of such cases, the choice of the three parsers is correct (identical to the gold standard). So the "agreement" accuracy of Mate+Turbo+MST versus Malt+Parsito is 62.8%. There are 7.000 cases when Mate, Turbo and MST agree, and Malt and Parsito each choose another dependency. In 61.2% of such cases, the choice of

the three parsers is correct. The "agreement" accuracy of Mate+Turbo+MST versus Malt and Parsito (not agreeing) is 61.2%.
Table 7 presents a part of the table recording the accuracy of "agreements" of parsers. Scores for unlabeled relations are presented (first column). The second column indicates which parsers agree on a dependency relation, the third column shows the agreement or disagreement of the other parsers. The fourth column shows the accuracy score, i. e. the ratio of correct dependency relations among all occurrences of this combination of agreements. The fourth column presents the number of occurrences. The table is sorted by accuracy.

Table 7: Accuracy of "agreements" of parsers

Un/lab	Parser(s)	Other parsers	Accuracy	Occurr.
UAS	all agree	none	97.16	867999
...	...	...	...	...
UAS	Malt+MST+Parsito	Mate+Turbo	47.25	5543
UAS	Mate+Turbo	Malt+Parsito | MST	46.96	2329
UAS	Mate+Turbo	Malt+MST | Parsito	45.71	1426
UAS	Mate+Turbo	Malt+MST+Parsito	44.70	5543
UAS	Mate+Turbo	Malt | MST | Parsito	44.53	2237
UAS	Mate+Turbo	Malt | MST+Parsito	44.34	1449

When using unlabeled dependency relations, any three parsers agreeing outperform any pair of parsers. With labeled dependencies, one pair of parsers, Mate+Parsito, has slightly better results when opposing the other three agreeing parsers.

5.2 Ensemble parsing using agreements of parsers

We sort agreements of parsers by their reliability and use this information in a combination of parsers. In new sentences (test data) parsed with all parsers, we detect for each token, which parsers agree and which disagree, and we choose for each token one dependency relation which has the highest "accuracy of agreement" value, for example, if Malt+MST+Parsito chooses one dependency relation for the given token and Mate+Turbo chooses another one, we choose the dependency indicated by the three parsers.
Should any cycle occur in the output of the combination of parsers, the algorithm assigns a new governing token to the member of the cycle with the lowest value of agreements of parsers.
Unlabeled and labeled reliability of agreements of parsers can be applied. If unlabeled scores are used, first the dependency relation is determined, then the dependency label is chosen amid the labels proposed by parsers which initially agreed on the dependency relation according to labeled agreement scores. If labeled scores are used, dependency relation and dependency label are treated together from the start.
Table 8 shows the results of both approaches (labeled and unlabeled agreement scores).

Table 8: Accuracy of combinations of parsers (dtest)

	Un/lab	UAS	LAS	SENT_U	SENT_L
Agreements	LAS	88.58	80.89	46.07	29.62
Agreements	UAS	90.11	80.55	47.67	28.53

The procedure using unlabeled accuracy scores of agreements of parsers has better results in UAS, and the difference between the LAS scores is low. The approach using unlabeled agreements has very good unlabeled results (UAS, SENT_U), but comparatively poor labeled results (LAS, SENT_L).

6 Results

In this section, we summarize the results of our experiments, we present our baseline and an hypothetical ceiling, and we discuss parsing speed.

6.1 Etest results

As the baseline for our results, we use the accuracy of MateParser which has a slightly lower UAS than TurboParser, but its labeled scores are far better. We calculate a hypothetical floor and ceiling for the accuracy of the combination of our five parsers (see 3.2). Table 9 shows the results of our two experiments with ensemble parsing. UAS, LAS, SENT_U and SENT_L scores are presented. The best settings for our ensemble parsing methods (tuned up on the dtest data) were tested on PDT etest data.

Table 9: Accuracy of combinations of parsers (etest)

	UAS	LAS	SENT_U	SENT_L
Floor	74.52	65.33	26.54	17.61
MateParser	88.58	83.09	45.87	33.77
Error rates	90.04	84.77	47.57	34.70
Agreements	90.07	81.91	47.61	30.65
Ceiling	95.51	92.18	69.36	55.71

A 1.5% improvement in UAS and a 1.7% improvement in SENT_U (unlabeled attachment score for the whole sentences) compared to the baseline was achieved by both ensemble parsing methods. As for labeled scores, the approach using error rates attained a 1.7% LAS and a 0.9% SENT_L improvement, whereas the method using agreements of parsers has worse labeled results than the baseline (but better than the average of the parsers). The reason for this difference lies probably in the more sophisticated way in which dependency labels are chosen by the method with error rates, which reflects better the strengths of the parsers in the domain of dependency labels. The method

of dealing with cycles in the experiment with the agreements of parsers is perhaps also to blame, in the future, we plan to use a maximum spanning tree algorithm, too.

6.2 Speed

We claimed in the introduction that parsing speed is not important for some tasks in NLP, such as corpus annotation. It can still be an issue when the data to be parsed are large, even if most of the process can be parallelized.
We measured the speed of all five parsers and of the program handling the combination of parsers on an Intel Xeon E5-2670 2,3GHz machine on the PDT etest data (approx. 8.000 sentences), using a single thread mode. Table 10 shows both speed (in sentences per second) and parsing time (in seconds per sentence) for the five parsers we used and for our ensemble parsing tools.

Table 10: Speed of parsers and ensemble parsing tools

Parser	Speed (sent/s)	Parsing time (s/sent)
MateParser	2.08	0.48
TurboParser	8.33	0.12
MaltParser	0.47	2.12
Parsito	16.67	0.06
MSTParser	11.11	0.09
Ensemble error rates	25	0.04
Ensemble agreements	50	0.02

The speed of the whole process of ensemble parsing in our experiments was determined by the speed of the slowest parser (MaltParser), which needs 3x more time per sentence than all the other parsers together. The merging of outputs of parsers and their combination (a perl program) requires only a negligible amount of time. Excluding the slowest parser would increase parsing speed considerably, but it would significantly decrease parsing accuracy (only 0.2% UAS, but almost 1.0% SENT_L), it would be therefore better to try to replace MaltParser by another parser, faster, but with good results in ensemble parsing. MaltParser trained with *liblinear* algorithm instead of *libsvm* is faster, but with far worse results in parsing PDT data.

6.3 Applicability to other languages

We did not test our approach on other languages because of a lack of time and computational resources, we intend to do that in the future. The most important points in the procedure are: optimize four or five parsers, parse training data using 10-fold cross-validation, gather information about the behavior and quality of the parsers from parsed training data. Then train all parsers again using training data and parse train data, merge parsing results and use the previously gathered information (using morphosyntactic parameters or agreements of parsers) in ensemble parsing as weights using an algorithm for finding a maximum spanning tree.
A 10-fold cross-validation over the whole data is also possible, but it would require a great amount of computational resources, as it would necessitate 110 cycles of training and parsing multiplied by the number of the parsers used.

7 Conclusion

In this paper, we have presented two methods of ensemble parsing which both achieve a significant (1.4%) increase in unlabeled attachment score compared to the best parser used. The approach using error rates calculated for each parser as weights in a combination of parsers using an algorithm for finding the maximum spanning tree in an oriented graph attains also very good labeled scores (1.7% increase in LAS).

References

[1] B. Bohnet, J. Nivre, "A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing," in Proceedings of EMNLP 2012, 2012.

[2] N.D. Green, Improvements to Syntax-based Machine Translation using Ensemble Dependency Parsers (thesis). Faculty of Mathematics and Physics, Charles University, Prague, 2013.

[3] N.D. Green, Z. Žabokrtský, "Hybrid combination of constituency and dependency trees into an ensemble dependency parser" in Proceedings of ACL 2012, 2012.

[4] J. Hajič, "Complex Corpus Annotation: The Prague Dependency Treebank," in Šimková M. (ed.): Insight into the Slovak and Czech Corpus Linguistics, pp. 54–73. Veda, Bratislava, Slovakia, 2006.

[5] T. Jelínek, "Improving Dependency Parsing by Filtering Linguistic Noise," in Proceedings of TSD 2013, 2013.

[6] A.F.T. Martins, M.B. Almeida, N.A. Smith, "Turning on the Turbo: Fast Third-Order Non-Projective Turbo Parsers," in Proceedings of ACL 2013, 2013.

[7] R. McDonald, F. Pereira, K. Ribarov, J. Hajic, "Non-projective Dependency Parsing using Spanning Tree Algorithms," in Proceedings of EMNLP 2005, 2005.

[8] J. Nivre, J. Hall, J. Nilsson, "MaltParser: A Data-Driven Parser-Generator for Dependency Parsing," in Proceedings of LREC 2006, 2006.

[9] M. Straka, J. Hajič, J. Straková, J. Hajič jr., "Parsing Universal Dependency Treebanks using Neural Networks and Search-Based Oracle," in Proceedings of TLT 2015, 2015.

[10] D. Zeman, Z. Žabokrtský, "Improving Parsing Accuracy by Combining Diverse Dependency Parsers," in Proceedings of IWPT 2005, 2005.