Appears in: Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-05), pp. 171-178, October 9-10, 2005, Association for Computational Linguistics, Vancouver, British Columbia, Canada

Improving Parsing Accuracy by Combining Diverse Dependency Parsers

Daniel Zeman and **Zdeněk Žabokrtský**
Ústav formální a aplikované lingvistiky, Univerzita Karlova
Malostranské náměstí 25, CZ-11800 Praha
{zeman|zabokrtsky}@ufal.mff.cuni.cz

Abstract

This paper explores the possibilities of improving parsing results by combining outputs of several parsers. To some extent, we are porting the ideas of Henderson and Brill (1999) to the world of dependency structures. We differ from them in exploring context features more deeply. All our experiments were conducted on Czech but the method is language-independent. We were able to significantly improve over the best parsing result for the given setting, known so far. Moreover, our experiments show that even parsers far below the state of the art can contribute to the total improvement.

1. Introduction

Difficult and important NLP problems have the property of attracting whole range of researchers, which often leads to the development of several different approaches to the same problem. If these approaches are independent enough in terms of not producing the same kinds of errors, there is a hope that their combination can bring further improvement to the field. While improving any single approach gets more and more difficult once some threshold has been touched, exploring the potential of approach combination should never be omitted, provided three or more approaches are available.

Combination techniques have been successfully applied to part of speech tagging (van Halteren et al., 1998; Brill and Wu, 1998; van Halteren et al., 2001). In both cases the investigators were able to achieve significant improvements over the previous best tagging results. Similar advances have been made in machine translation (Frederking and Nirenburg, 1994), speech recognition (Fiscus, 1997), named entity recognition (Borthwick et al., 1998), partial parsing (Inui and Inui, 2000), word sense disambiguation (Florian and Yarowsky, 2002) and question answering (Chu-Carroll et al., 2003).

Brill and Hladká (Hajič et al., 1998) have first explored committee-based dependency parsing. However, they generated multiple parsers from a single one using bagging (Breiman, 1994). There have not been more sufficiently good parsers available. A successful application of voting and of a stacked classifier to constituent parsing followed in (Henderson and Brill, 1999). The authors have investigated two combination techniques (constituent voting and naïve Bayes), and two ways of their application to the (full) parsing: parser switching, and similarity switching. They were able to gain 1.6 constituent F-score, using their most successful technique.

In our research, we focused on dependency parsing. One of the differences against Henderson and Brill's situation is that a dependency parser has to assign exactly one governing node (parent word) to each word. Unlike the number of constituents in constituency-based frameworks, the number of dependencies is known in advance, the parser only has to assign a link (number 0 through *N*) to each word. In that sense, a dependency parser is similar to classifiers like POS taggers. Unless it deliberately fails to assign a parent to a word (or assigns

Parser	Author	Brief description	Accuracy	
			Tune	Test
ec	Eugene Charniak	A maximum-entropy inspired parser, home in constituency-based structures. English version described in Charniak (2000), Czech adaptation 2002 – 2003, unpublished.	83.6	85.0
mc	Michael Collins	Uses a probabilistic context-free grammar, home in constituency-based structures. Described in (Hajič et al., 1998; Collins et al., 1999).	81.7	83.3
zž	Zdeněk Žabokrtský	Purely rule-based parser, rules are designed manually, just a few lexical lists are collected from the training data. 2002, unpublished.	74.3	76.2
dz	Daniel Zeman	A statistical parser directly modeling syntactic dependencies as word bigrams. Described in (Zeman, 2004).	73.8	75.5
thr	Tomáš Holan	Three parsers. Two of them use a sort of push-down automata and differ from each other only in the way they process the sentence (left-to-right or right-to-left). Described in (Holan, 2004).	71.0	72.3
thl			69.5	70.3
thp			62.0	63.5

Table 1. A brief description of the tested parsers. Note that the Tune data is *not* the data used to train the individual parsers. Higher numbers in the right column reflect just the fact that the Test part is slightly easier to parse.

several alternate parents to a word), there is no need for precision & recall. Instead, a single metric called accuracy is used.

On the other hand, a dependency parser is not a *real* classifier: the number of its "classes" is theoretically unlimited (natural numbers), and no generalization can be drawn about objects belonging to the same "class" (words that – sometimes – appeared to find their parent at the position *i*).

A combination of dependency parsers does not necessarily grant the resulting dependency structure being cycle-free. (This contrasts to not introducing crossing brackets in constituent parsing, which is granted according to Henderson and Brill.) We address the issue in 4.4.

The rest of this paper is organized as follows: in Sections 2 and 3 we introduce the data and the component parsers, respectively. In Section 4 we discuss several combining techniques, and in Section 5 we describe the results of the corresponding experiments. We finally compare our results to the previous work and conclude.

2. The data

To test our parser combination techniques, we use the Prague Dependency Treebank 1.0 (PDT; Hajič et al. 2001). All the individual parsers have been trained on its analytical-level training section (73,088 sentences; 1,255,590 tokens).

The PDT analytical d-test section has been partitioned into two data sets, Tune (last 77 files; 3646 sentences; 63,353 tokens) and Test (first 76 files; 3673 sentences; 62,677 tokens). We used the Tune set to train the combining classifiers if needed. The Test data were used to evaluate the approach. Neither the member parsers, nor the combining classifier have seen this data set during their respective learning runs.

3. Component parsers

The parsers involved in our experiments are summarized in Table 1. Most of them use unique strategies, the exception being *thl* and *thr*, which differ only in the direction in which they process the sentence.

The table also shows individual parser accuracies on our Test data. There are two state-of-the art parsers, four not-so-good parsers, and one quite poor parser. We included the two best parsers (ec+mc) in all our experiments, and tested the contributions of various selections from the rest.

The necessary assumption for a meaningful combination is that the outputs of the individual parsers are sufficiently uncorrelated, i.e. that the parsers do not produce the same errors. If some

Parsers compared		**All 7**	**4 best**	**3 best**	**ec+mc+dz**	**2 best**	**3 worst**
Who is correct		**How many times correct**					
a single parser (all other wrong)	ec	1.7 %	3.0 %	4.1 %	4.5 %	8.1 %	
	zž	1.2 %	2.0 %	3.3 %			
	mc	0.9 %	1.7 %	2.7 %	2.9 %	6.2 %	
	thr	0.4 %					4.9 %
	thp	0.4 %					4.4 %
	dz	0.3 %	1.0 %		2.2 %		
	thl	0.3 %					4.3 %
all seven parsers		42.5 %					
at least six		58.1 %					
at least five		68.4 %					
at least four		76.8 %	58.0 %				
at least three		84.0 %	75.1 %	63.6 %	64.7 %		50.6 %
at least two		90.4 %		82.9 %	82.4 %	75.5 %	69.2 %
at least one		95.8 %	94.0 %	93.0 %	92.0 %	89.8 %	82.7 %

Table 2: Comparison of various groups of parsers. All percentages refer to the share of the total words in test data, attached correctly. The "single parser" part shows shares of the data where a single parser is the only one to know how to parse them. The sizes of the shares should correlate with the uniqueness of the individual parsers' strategies and with their contributions to the overall success. The "at least" rows give clues about what can be got by majority voting (if the number represents over 50 % of parsers compared) or by hypothetical oracle selection (if the number represents 50 % of the parsers or less, an oracle would generally be needed to point to the parsers that know the correct attachment).

parsers produced too similar results, there would be the danger that they push all their errors through, blocking any meaningful opinion of the other parsers.

To check the assumption, we counted (on the Tune data set) for each parser in a given parser selection the number of dependencies that only this parser finds correctly. We show the results in Table 2. They demonstrate that all parsers are independent on the others at least to some extent.

4. Combining techniques

Each dependency structure consists of a number of dependencies, one for each word in the sentence. Our goal is to tell for each word, which parser is the most likely to pick its dependency correctly. By combining the selected dependencies we aim at producing a better structure. We call the complex system (of component parsers plus the selector) the *superparser*.

Although we have shown how different strategies lead to diversity in the output of the parsers, there is little chance that any parser will be able to push through the things it specializes in. It is very difficult to realize that a parser is right if most of the others reject its proposal. Later in this section we assess this issue; however, the real power is in majority of votes.

4.1. Voting

The simplest approach is to let the member parsers vote. At least three parsers are needed. If there are exactly three, only the following situations really matter: 1) two parsers outvote the third one; 2) a tie: each parser has got a unique opinion. It would be democratic in the case of a tie to select randomly. However, that hardly makes sense once we know the accuracy of the involved parsers on the Tune set. Especially if there is such a large gap between the parsers' performance, the best parser (here ec) should get higher priority whenever there

is no clear majority of votes. Van Halteren et al. (1998) have generalized this approach for higher number of classifiers in their TotPrecision voting method. The vote of each classifier (parser) is weighted by their respective accuracy. For instance, mc + zž would outvote ec + thr, as 81.7 + 74.3 = 156 > 154.6 = 83.6 + 71.0.

4.2. Stacking

If the world were ideal, we would have an oracle, able to *always* select the right parser. In such situation our selection of parsers would grant the accuracy as high as 95.8 %. We attempt to imitate the oracle by a second-level classifier that learns from the Tune set, which parser is right in which situations. Such technique is usually called *classifier stacking*. Parallel to (van Halteren et al., 1998), we ran experiments with two stacked classifiers, Memory-Based, and Decision-Tree-Based. This approach roughly corresponds to (Henderson and Brill, 1999)'s Naïve Bayes parse hybridization.

4.3. Unbalanced combining

For applications preferring precision to recall, unbalanced combination — introduced by Brill and Hladká in (Hajič et al., 1998) — may be of interest. In this method, all dependencies proposed by at least half of the parsers are included. The term *unbalanced* reflects the fact that now precision is *not* equal to recall: some nodes lack the link to their parents. Moreover, if the number of member parsers is even, a node may get two parents.

4.4. Switching

Finally, we develop a technique that considers the whole dependency structure rather than each dependency alone. The aim is to check that the resulting structure is a tree, i.e. that the dependency-selecting procedure does not introduce cycles.[1] Henderson and Brill prove that under certain conditions, their parse hybridization approach cannot introduce crossing brackets. This might seem an analogy to our problem of introducing cycles — but unfortunately, no analogical lemma holds. As a workaround, we have investigated a crossbreed approach between Henderson and Brill's Parser Switching, and the voting methods described above. After each step, all dependencies that would introduce a cycle are banned. The algorithm is greedy — we do not try to search the space of dependency combinations for other paths. If there are no allowed dependencies for a word, the whole structure built so far is abandoned, and the structure suggested by the best component parser is used instead.[2]

5. Experiments and results

5.1. Voting

We have run several experiments where various selections of parsers were granted the voting right. In all experiments, the TotPrecision voting scheme of (van Halteren et al., 1998) has been used. The voting procedure is only very moderately affected by the Tune set (just the accuracy figures on that set are used), so we present results on both the Test and the Tune sets.

Voters	Accuracy	
	Tune	Test
ec (baseline)	83.6	85.0
all seven	84.0	85.4
ec+mc+dz	84.9	86.2
all but thp	84.9	86.3
ec+mc+zž+dz+thr	85.1	86.5
ec+mc+zž	85.2	86.7
ec+mc+zž+dz	85.6	87.0

Table 3: Results of voting experiments.

According to the results, the best voters pool consists of the two best parsers, accompanied by

[1] One may argue that "treeness" is not a necessary condition for the resulting structure, as the standard accuracy measure does not penalize non-trees in any way (other than that there is at least one bad dependency). Interestingly enough, even some of the component parsers do not produce correct trees at all times. However, non-trees are both linguistically and technically problematic, and it is good to know how far we can get with the condition in force.

[2] We have not encountered such situation in our test data. However, it indeed is possible, even if all the component parsers deliver correct trees, as can be seen from the following example. Assume we have a sentence #ABCD and parsers P1 (85 votes), P2 (83 votes), P3 (76 votes). P1 suggests the tree A→D→B→C→#, P2 suggests B→D→A→C→#, P3 suggests B→D→A→#, C→#. Then the superparser P gradually introduces the following dependencies: 1. A→D; 2. B→D; 3. C→#; 4. D→A or D→B possible but both lead to a cycle.

the two average parsers. The table also suggests that number of diverse strategies is more important than keeping high quality standard with all the parsers. Apart from the worst parser, all the other together do better than just the first two and the fourth. (On the other hand, the first three parsers are much harder to beat, apparently due to the extreme distance of the strategy of zž parser from all the others.)

Even the worst performing parser combination (all seven parsers) is significantly[3] better than the best component parser alone.

We also investigated some hand-invented voting schemes but no one we found performed better than the ec+mc+zž+dz combination above.

Some illustrative results are given in the Table 4. Votes were not weighted by accuracy in these experiments, but accuracy is reflected in the priority given to ec and mc by the human scheme inventor.

Voters	Selection scheme	Accuracy	
		Tune	Test
all seven	most votes or ec	82.8	84.3
all seven	at least half, or ec if there is no absolute majority	84.4	85.8
all seven	absolute majority, or ec+2, or mc+2, or ec	84.6	85.9

Table 4: Voting under hand-invented schemes.

5.2. Stacking – using context

We explored several ways of using context in pools of three parsers.[4] If we had only three parsers we could use context to detect two kinds of situations:

1. Each parser has its own proposal and a parser other than ec shall win.
2. Two parsers agree on a common proposal but even so the third one should win. Most likely the only reasonable instance is that ec wins over mc + the third one.

"Context" can be represented by a number of features, starting at morphological tags and ending up at complex queries on structural descriptions. We tried a simple memory-based approach, and a more complex approach based on decision trees.

Within the memory-based approach, we use just the core features the individual parsers themselves train on: the POS tags (morphological tags or m-tags in PDT terminology). We consider the m-tag of the dependent node, and the m-tags of the governors proposed by the individual parsers.

We learn the context-based strengths and weaknesses of the individual parsers on their performance on the Tune data set. In the following table, there are some examples of contexts in which ec is better than the common opinion of mc + dz.

Dep. tag	Gov. tag (ec)	Context occurrences	No. of times ec was right	Percent cases ec was right
J^	#	67	44	65.7
Vp	J^	53	28	52.8
VB	J^	46	26	56.5
N1	Z,	38	21	55.3
Rv	Vp	25	13	52.0
Z,	Z,	15	8	53.3
A1	N1	15	8	53.3
Vje	J^	14	9	64.3
N4	Vf	12	9	75.0

Table 5: Contexts where ec is better than mc+dz. J^ are coordination conjunctions, # is the root, V* are verbs, N*n* are nouns in case *n*, R* are prepositions, Z* are punctuation marks, A*n* are adjectives.

For the experiment with decision trees, we used the C5 software package, a commercial version of the well-known C4.5 tool (Quinlan, 1993). We considered the following features:

For each of the four nodes involved (the dependent and the three governors suggested by the three component parsers):

[3] All significance claims refer to the Wilcoxon Signed Ranks Test at the level of $p = 0.001$.

[4] Similar experiments could be (and have been) run for sets of more parsers as well. However, the number of possible features is much higher and the data sparser. We were not able to gain more accuracy on context-sensitive combination of more parsers.

```
agreezzmc = yes: zz (3041/1058)
agreezzmc = no:
:...agreemcec = yes: ec (7785/1026)
    agreemcec = no:
    :...agreezzec = yes: ec (2840/601)
        agreezzec = no:
        :...zz_case = 6: zz (150/54)
            zz_case = 3: zz (34/10)
            zz_case = X: zz (37/20)
            zz_case = undef: ec (2006/1102)
            zz_case = 7: zz (83/48)
            zz_case = 2: zz (182/110)
            zz_case = 4: zz (108/57)
            zz_case = 1: ec (234/109)
            zz_case = 5: mc (1)
            zz_case = root:
            :...ec_negat = A: mc (117/65)
                ec_negat = undef: ec (139/65)
                ec_negat = N: ec (1)
                ec_negat = root: ec (2)
```

Figure 1. The decision tree for ec+mc+zž, learned by C5. Besides pairwise agreement between the parsers, only morphological case and negativeness matter.

- 12 attributes derived from the morphological tag (part of speech, subcategory, gender, number, case, inner gender, inner number, person, degree of comparison, negativeness, tense and voice)
- 4 semantic attributes (such as ProperName, Geography etc.)

For each of the three governor-dependent pairs involved:

- mutual position of the two nodes (LeftNeighbor, RightNeighbor, LeftFar, RightFar)
- mutual position expressed numerically
- for each parser pair a binary flag whether they do or do not share opinions

The decision tree was trained only on situations where at least one of the three parsers was right *and* at least one was wrong.

Voters	Scheme	Accuracy
ec+mc+dz	context free	86.2
ec+mc+dz	memory-based	86.3
ec+mc+zž	context free	86.7
ec+mc+zž	decision tree	86.9

Table 6: Context-sensitive voting. Contexts trained on the Tune data set, accuracy figures apply to the Test data set. Context-free results are given for the sake of comparison.

It turns out that there is very low potential in the context to improve the accuracy (the improvement is significant, though). The behavior of the parsers is too noisy as to the possibility of formulating some rules for prediction, when a particular parser is right. C5 alone provided a supporting evidence for that hypothesis, as it selected a very simple tree from all the features, just 5 levels deep (see Figure 1).

Henderson and Brill (1999) also reported that context did not help them to outperform simple voting. Although it is risky to generalize these observations for other treebanks and parsers, our environment is quite different from that of Henderson and Brill, so the similarity of the two observations is at least suspicious.

5.3. Unbalanced combining

Finally we compare the balanced and unbalanced methods. Expectedly, precision of the unbalanced combination of odd number of parsers rose while recall dropped slightly. A different situation is observed if even number of parsers vote and more than one parent can be selected for a node. In such case, precision drops in favor of recall.

Method	Precision	Recall	F-measure
ec only (baseline)	85.0		
balanced (all seven)	85.4		
unbalanced (all seven)	**90.7**	78.6	84.2
balanced (best four)	**87.0**		
unbalanced (best four)	85.4	**87.7**	86.5
balanced (ec+mc+dz)	86.2		
unbalanced	89.5	84.0	86.7

Method	Precision	Recall	F-measure
(ec+mc+dz)			
balanced (ec+mc+zž)	86.7		
unbalanced (ec+mc+zž)	90.2	84.7	**87.3**

Table 7: Unbalanced vs. balanced combining. All runs ignored the context. Evaluated on the Test data set.

5.4. Switching

Out of the 3,673 sentences in our Test set, 91.6 % have been rendered as correct trees in the balanced decision-tree based stacking of ec+mc+zž+dz (our best method).

After we banned cycles, the accuracy dropped from 87.0 to 86.9 %.[5]

6. Comparison to related work

Brill and Hladká in (Hajič et al., 1998) were able to improve the original accuracy of the mc parser on PDT 0.5 e-test data from 79.1 to 79.9 (a nearly 4% reduction of the error rate). Their unbalanced[6] voting pushed the F-measure from 79.1 to 80.4 (6% error reduction). We pushed the balanced accuracy of the ec parser from 85.0 to 87.0 (13% error reduction), and the unbalanced F-measure from 85.0 to 87.7 (18% reduction). Note however that there were different data and component parsers (Hajič et al. found bagging the best parser better than combining it with other that-time-available parsers). This is the first time that several strategically different dependency parsers have been combined.

(Henderson and Brill, 1999) improved their best parser's F-measure of 89.7 to 91.3, using their naïve Bayes voting on the Penn TreeBank constituent structures (16% error reduction). Here, even the framework is different, as has been explained above.

7. Conclusion

We have tested several approaches to combining of dependency parsers. Accuracy-aware voting of the four best parsers turned out to be the best method, as it significantly improved the accuracy of the best component from 85.0 to 87.0 % (13 % error rate reduction). The unbalanced voting lead to the precision as high as 90.2 %, while the F-measure of 87.3 % outperforms the best result of balanced voting (87.0).

At the same time, we found that employing context to this task is very difficult even with a well-known and widely used machine-learning approach.

The methods are language independent, though the amount of accuracy improvement may vary according to the performance of the available parsers.

Although voting methods are themselves not new, as far as we know we are the first to propose and evaluate their usage in full dependency parsing.

8. Acknowledgements

Our thanks go to the creators of the parsers used here for making their systems available.

The research has been supported by the Czech Academy of Sciences, the "Information Society" program, project No. 1ET101470416.

References

Andrew Borthwick, John Sterling, Eugene Agichtein, Ralph Grishman. 1998. *Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition.* In: Eugene Charniak (ed.): Proceedings of the 6th Workshop on Very Large Corpora, pp. 152–160. Université de Montréal, Montréal, Québec.

Leo Breiman. 1994. *Bagging Predictors.* Technical Report 421, Department of Statistics, University of California at Berkeley, Berkeley, California.

Eric Brill, Jun Wu. 1998. *Classifier Combination for Improved Lexical Combination.* In: Proceedings of the 17th International Conference on Computational Linguistics (COLING-98), pp. 191–195. Université de Montréal, Montréal, Québec.

Eugene Charniak. 2000. *A Maximum-Entropy-Inspired Parser.* In: Proceedings of NAACL. Seattle, Washington.

Jennifer Chu-Carroll, Krzysztof Czuba, John Prager, Abraham Ittycheriah. 2003. *In Question Answering, Two Heads Are Better Than One.* In: Proceedings of the HLT-NAACL. Edmonton, Alberta.

[5] The authors apologize for the typo in the version published in the IWPT proceedings ("97.0 to 96.9 %", instead of "87.0 to 86.9 %").

[6] Also alternatively called *unrestricted*.

Michael Collins, Jan Hajič, Eric Brill, Lance Ramshaw, Christoph Tillmann. 1999. *A Statistical Parser of Czech.* In: Proceedings of the 37th Meeting of the ACL, pp. 505–512. University of Maryland, College Park, Maryland.

Jonathan G. Fiscus. 1997. *A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER).* In: EuroSpeech 1997 Proceedings, vol. 4, pp. 1895–1898. Rodos, Greece.

Radu Florian, David Yarowsky. 2002. *Modeling Consensus: Classifier Combination for Word Sense Disambiguation.* In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 25–32. Philadelphia, Pennsylvania.

Robert Frederking, Sergei Nirenburg. 1994. *Three Heads Are Better Than One.* In: Proceedings of the 4th Conference on Applied Natural Language Processing, pp. 95–100. Stuttgart, Germany.

Jan Hajič, Eric Brill, Michael Collins, Barbora Hladká, Douglas Jones, Cynthia Kuo, Lance Ramshaw, Oren Schwartz, Christoph Tillmann, Daniel Zeman. 1998. *Core Natural Language Processing Technology Applicable to Multiple Languages. The Workshop 98 Final Report.* http://www.clsp.jhu.edu/ws98/projects/nlp/report/. Johns Hopkins University, Baltimore, Maryland.

Jan Hajič, Barbora Vidová Hladká, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas. 2001. *Prague Dependency Treebank 1.0 CD-ROM.* Catalog # LDC2001T10, ISBN 1-58563-212-0. Linguistic Data Consortium, Philadelphia, Pennsylvania.

Hans van Halteren, Jakub Zavřel, Walter Daelemans. 1998. *Improving Data-Driven Wordclass Tagging by System Combination.* In: Proceedings of the 17th International Conference on Computational Linguistics (COLING-98), pp. 491–497. Université de Montréal, Montréal, Québec.

Hans van Halteren, Jakub Zavřel, Walter Daelemans. 2001. *Improving Accuracy in Word Class Tagging through the Combination of Machine Learning Systems.* In: Computational Linguistics, vol. 27, no. 2, pp. 199–229. MIT Press, Cambridge, Massachusetts.

John C. Henderson, Eric Brill. 1999. *Exploiting Diversity in Natural Language Processing: Combining Parsers.* In: Proceedings of the Fourth Conference on Empirical Methods in Natural Language Processing (EMNLP-99), pp. 187–194. College Park, Maryland.

Tomáš Holan. 2004. *Tvorba závislostního syntaktického analyzátoru.* In: David Obdržálek, Jana Tesková (eds.): MIS 2004 Josefův Důl, Sborník semináře. Matfyzpress, Praha, Czechia.

Inui Takashi, Inui Kentaro. 2000. *Committee-Based Decision Making in Probabilistic Partial Parsing.* In: Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000), pp. 348–354. Universität des Saarlandes, Saarbrücken, Germany.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California.

Daniel Zeman. 2004. *Parsing with a Statistical Dependency Model (PhD thesis).* Univerzita Karlova, Praha, Czechia.