Improving Dependency Parsing Using Sentence Clause Charts

Vincent Kríž and Barbora Hladká
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{kriz, hladka}@ufal.mff.cuni.cz

Abstract

We propose a method for improving the dependency parsing of complex sentences. This method assumes segmentation of input sentences into clauses and does not require to re-train a parser of one's choice. We represent a sentence clause structure using clause charts that provide a layer of embedding for each clause in the sentence. Then we formulate a parsing strategy as a two-stage process where (i) coordinated and subordinated clauses of the sentence are parsed separately with respect to the sentence clause chart and (ii) their dependency trees become subtrees of the final tree of the sentence. The object language is Czech and the parser used is a maximum spanning tree parser trained on the Prague Dependency Treebank. We have achieved an average 0.97% improvement in the unlabeled attachment score. Although the method has been designed for the dependency parsing of Czech, it is useful for other parsing techniques and languages.

1 Introduction

Syntactic parsing is an integral part of a complex text processing pipeline whose quality impacts the overall performance of the text processing system.

For illustration, we share our experience with a system focused on extracting semantic relations from unstructured texts. Namely, we have developed the RExtractor system,[1] which processes texts by linguistically-aware tools and extracts entities and relations using queries over dependency trees. The language used for testing RExtractor was Czech and the legal domain was chosen to be explored in detail (Kríž et al., 2014; Kríž and Hladká, 2015). We evaluated RExtractor on the Czech Legal Text Treebank (CLTT) enriched with manually annotated entities and their relations (Kríž et al., 2016). Because of the lack of any Czech gold legal-domain treebank, we used a parser trained on newspaper texts to parse CLTT. The RExtractor system achieved precision of 80.6% and recall of 63.2% and we identified three sources of errors: (i) incorrect dependency tree (59.7%), (ii) missing or incorrectly formulated extraction query (38.3%), (iii) missing or incorrectly recognized entity (2.1%).

One can see that the errors are caused mainly by the insufficient quality of dependency parsing. The main reason why it happens is that newspaper texts differ from legal texts in several language phenomena influenced by the high frequency of very long sentences in legal texts. Figure 1 provides evidence of difficulty with dependency parsing long sentences – as the sentence length increases, the unlabeled attachment score decreases. The numbers are provided for two Czech dependency treebanks, namely the Prague Dependency Treebank with the development and evaluation test subsets[2] (PDT, dtest, etest, resp.) and the Czech Academic Corpus (CAC)[3], see Bejček et al. (2013) and Hladká et al. (2008), resp.

This paper describes our method how to use information about a sentence clause structure in full-scale dependency parsing. Section 2 lists a number of previous approaches to improve dependency parsing including selected works on parsing Czech. The data and tools used in our experiments are summarized in Section 3. We represent sentence clause structures using clause charts defined and quantitatively studied in Section 4. In

[1]The system is available on-line at http://quest.ms.mff.cuni.cz:14280/

[2]https://ufal.mff.cuni.cz/pdt3.0

[3]https://ufal.mff.cuni.cz/cac

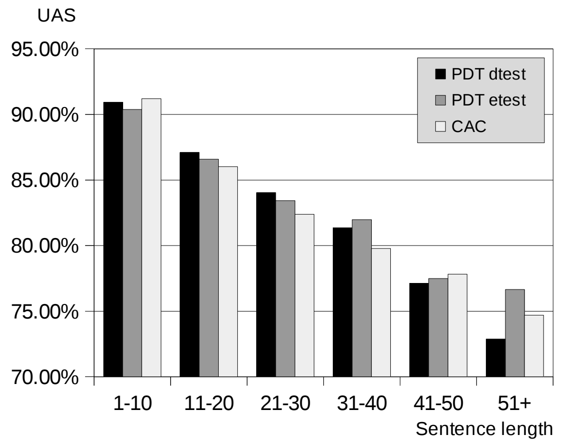


Figure 1: The longer the sentence the lower the unlabeled attachment score. The figures come from experiments run on the Prague Dependency Treebank and the Czech Academic Corpus.

Section 5, we propose an original strategy to parse pairs of coordinated and subordinated clauses and apply it on the data. Section 6 outlines our future plans towards better parsing long sentences.

2 Related Work

Several approaches which deal with the idea of dividing the parsing process into several parts were presented. The idea of cascaded parsing exploits a cascade of specialized parsers instead of having one very complex general parser (Abney, 1996; Ciravegna and Lavelli, 1999). The identification of chunks, syntactically related non-overlapping groups of words (Tjong Kim Sang and Buchholz, 2000), was used mainly in shallow parsing strategies (Federici et al., 1996). Clausal parsing was designed to parse Hindi texts (Husain et al., 2011).

However, there is no work on exploiting chunks for full-scale parsing. A very interesting approach dividing the parsing process into several parts has been introduced in the XDG theory (Debusmann et al., 2005). Most recent approaches to dependency parsing focus almost exclusively on improving full-scale parsing algorithms using mostly neural networks (Pei et al., 2015; Weiss et al., 2015; Zhu et al., 2015).

We address the issue of parsing sentences that are already segmented into clauses. The ideas and concepts of segmentation of Czech sentences are presented by Kuboň (2001), Kuboň et al. (2007), and Lopatková and Holan (2009). They present the concept of segments and show that it is possible to draw the segmentation charts which reflect the mutual position of segments in complex sentences without applying syntactic parsing of the whole sentence first. The method is based on the identification of separators (segment boundaries) and their classification.

Lopatková et al. (2012) show how clauses forming complex sentences can be identified based on the sentence segment annotation. In addition, they present the project aiming at building a collection of Czech sentences enriched with manually annotated clauses and their relationships. Krůza and Kuboň (2014) use this collection to develop an automatic procedure for recognizing clauses and their mutual relationship. Another automatic procedure for clause identification over dependency trees is introduced by Bejček et al. (2013) achieved F-measure 97.51% and was used for the clause annotation of the Prague Dependency Treebank.

3 Data and Tools

We experimented with two manually annotated dependency treebanks, namely the Prague Dependency Treebank 3.0 (PDT 3.0) and the Czech Academic Corpus 2.0 (CAC 2.0). Both corpora are enriched with the clause annotation done automatically using the procedure presented by Bejček et al. (2013).

Our goal is to beat the MST dependency parser (McDonald et al., 2005) trained on the PDT 3.0 train set. Table 1 presents basic characteristics of the two treebanks and the MST parser performance on them.

Treebank	Split	Sent.	Tokens	UAS
PDT 3.0	train	29,768	518,648	93.41
	dtest	4,042	70,974	84.50
	etest	4,672	80,923	84.32
CAC 2.0	–	24,709	493,306	82.68

Table 1: Part of the treebank (*Split*), the number of sentences (*Sent.*), the number of tokens (*Tokens*) and the unlabeled attachment score (*UAS*) of MST.

4 Clause Charts

A *clause chart* is defined to visualize relationships between clauses within the sentence and captures the layer of embedding of each individual clause. It is an $m \times n$ table where n is the number of clauses in the sentence and m is the number of layers. A cell (i, j) stands for relationship between the j-th clause and the i-th layer of embedding. Its value is initialized to the value of 0 corresponding to no relationship.

We defined four rules for changing the cell value from 0 to 1, i.e., for assigning a layer of embedding to each clause in the sentence: (1) All main clauses belong to the basic layer 0. (2) The clauses that depend on the clauses at the k-th layer belong to the $(k+1)$-th layer. (3) The coordinated clauses and the clauses in apposition belong to the same layer. (4) The clauses in parentheses belong to the $(k+1)$-th layer with respect to the k-th layer of their adjacent clauses.

Our definition is analogous to a segmentation chart defined by Lopatková and Holan (2009). However, we handle the following situations differently: (1) subordinating conjunctions at the beginning of each clause are considered as boundaries and are excluded from the clause; (2) clauses split into two parts by an embedded subordinated clause are considered as two different clauses.

4.1 Generating Clause Charts

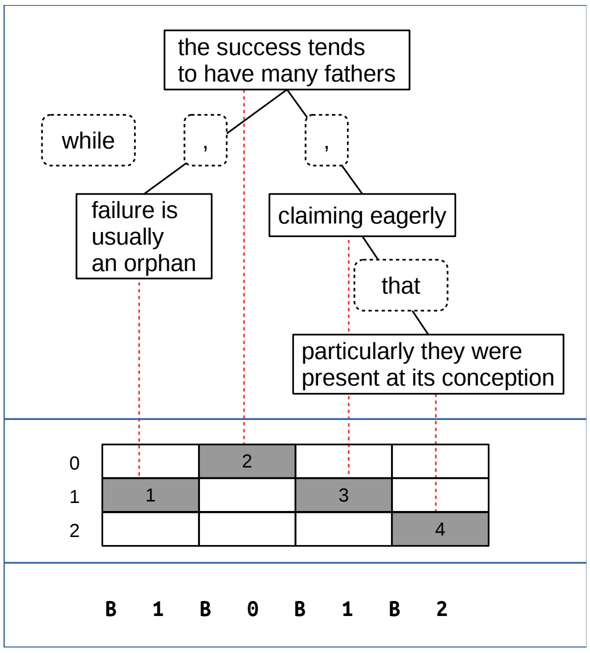
We designed a simple procedure that generates a clause chart from a dependency tree with the clause annotation. Particularly, it generates a clause tree first and then a clause chart.

We assume a dependency tree where each non-boundary node has a special attribute bearing the identification of the clause it belongs to. The nodes with the same clause number belong to the same clause and thus generating a clause chart is uniquely determined by the clause identification. A layer of embedding of the clause is defined as its depth in a sentence clause tree where its nodes contain tokens with the same clause identification.

Figure 2 displays both the clause tree and the clause chart of the sample sentence presented in (Kuboň et al., 2007):

> ***While*** *failure is usually an orphan****,*** *the success tends to have many fathers****,*** *claiming eagerly* ***that*** *particularly they were present at its conception****.***

This sentence consists of four clauses delimited by the boundaries printed in bold, namely *while*, *that*, and two commas. In general, clause boundaries are either a single token or a sequence of tokens. Clause boundaries are not components of the clause tree. They are displayed there for



the success tends to have many fathers
while
,
,
failure is usually an orphan
claiming eagerly
that
particularly they were present at its conception
0
1
2
2
1
3
4
B 1 B 0 B 1 B 2

Figure 2: Clause tree (above), clause chart and its linear representation (below).

understanding a linear representation of a clause chart, see `B1B0B1B2` where `B` stands for a clause boundary and the numbers are the layers of clause embedding.

4.2 Exploring Clause Charts

We explored PDT 3.0 and CAC 2.0 to study different types of clause charts. Table 2 provides statistics for the five most frequent clause charts that occur in the treebanks. For example, 14.4% of the sentences in PDT 3.0 and CAC 2.0 contain a main clause and a subordinated clause described with the `0B1` pattern. Moreover, we measure the MST parser performance on the sentences having the given clause charts. For example, MST achieved UAS of 92.9% on the `0B1B0` sentences in the PDT training data set.

The texts in the treebanks come from newspapers. Thus there is no surprise that the most frequent sentences in the treebanks are simple one clause sentences (`0`). They present more than half of the data. The second most frequent sentence structure consists of one main clause and one subordinated clause (`0B1`). It is quite surprising that the parser processes these sentences better than the one clause sentences. Even more, we observe decrease in the parser performance on coordination of two main clauses (i.e., on the `0B0` sentence).

For curiosity's sake, the most complex sentence in the treebanks consists of 36 clauses and the 0B1B2B3B4B5B6 clause chart is a chart with the highest number of layers of embedding.

	0	0B1	0B0	0B1B0	0B1B2
Rel. freq.	50.1	14.5	8.0	3.6	2.5
PDT train	93.6	95.8	92.9	92.9	95.9
PDT dtest	85.7	88.2	82.3	81.7	90.0
PDT etest	85.4	88.0	83.4	82.0	88.1
CAC 2.0	84.1	85.7	81.0	79.7	87.3

Table 2: Relative frequency of the five most frequent clause charts in PDT 3.0 and CAC 2.0 (*Rel. freq.*) and the unlabeled attachment score of MST evaluated on the particular subsets PDT train, PDT dtest, PDT etest, CAC 2.0.

5 Methods and Experiments

We present a method for improving dependency parsing of long sentences. In particular, we formulate an algorithm for parsing the two most frequent clause structures, namely coordinated clauses 0B0 and governing and dependent clauses 0B1. The other types of clause structures are processed as usual using full-scale parsing. The experiments exploit an existing dependency parser trained on complete sentences, namely the MST parser – see Section 3 for details.

5.1 Parsing Coordinated Clauses

Given the clause chart representation, we can recognize coordinated clauses in sentences in a straightforward way. Thus, we consider neighboring coordinated clauses C_1, C_2, …, C_n on the same layer ($n > 1$) and we propose the following parsing strategy that we call *clause chart parsing* (CCP):

1. Using MST parse C_1, C_2, …, C_n individually to get dependency trees T_1, T_2, …, T_n with the r_1, r_2, …, r_n root nodes, respectively.

2. Create a sequence $S = r_1\ B_{1,2}\ r_2\ B_{2,3} \ldots r_n$ where $B_{i,i+1}$ is a boundary between C_i and C_{i+1}.

3. Using MST parse the sequence S to get a dependency tree T_S.

4. Build a final dependency tree so that the trees $T_1, \ldots, T_n$ become subtree of T_S.

For illustration, we assume the sentence *John loves Mary and Linda hates Peter*. The sentence consists of two coordinated clauses C_1 = {*John loves Mary*}, C_2 = {*Linda hates Peter*} and one clause boundary $B_{1,2}$ = {*and*}. Therefore, the clause chart of the sentence is 0B0. In Step 1, C_1 and C_2 are parsed to get T_1 and T_2 with the root nodes r_1 = *loves* and r_2 = *hates*, resp. In Step 2, the sequence S = *loves and hates* is created. In Step 3, S is parsed to get T_S and, finally, in Step 4, T_1 and T_2 become subtrees of T_S.

We evaluated the proposed parsing strategy only on the sentences having the 0B0 clause chart, i.e., on the subsets of the treebank datasets. Table 3 presents the unlabeled attachment score achieved for

- full-scale parsing, i.e., parsing complete sentences using MST (*FS*)
- parsing individual clauses instead of parsing complete sentences, i.e., MST performance is measured on individual clauses (*Clauses*)
- full-scale parsing using the *CCP* strategy

We observe that parsing performance measured on complete sentences is the highest when parsing individual clauses. Using the CCP method we achieved an average 1.36% improvement in UAS.

Data	Sent.	FS	Clauses	CCP
PDT dtest	319	82.28	86.87	84.80
PDT etest	352	83.43	87.16	84.67
CAC 2.0	2,272	80.96	84.69	82.34

Table 3: Parsing evaluation on the 0B0 sentences of three different parsing strategies: full-scale parsing (*FS*) using MST, parsing individual clauses (*Clauses*), and full-scale parsing using CCP (*CCP*).

5.2 Parsing governing and dependent clauses

Table 4 presents the unlabeled attachment score achieved for full-scale parsing and parsing individual clauses.

We observe almost no improvement when parsing individual clauses. Also, we observe that the parser performance on the 0B1 sentences is significantly higher than the parser performance on the

Data	Sent.	FS	Clauses
PDT dtest	604	88.24	88.23
PDT etest	704	87.98	88.64
CAC 2.0	3,669	85.68	85.76

Table 4: Parsing evaluation on the `0B1` sentences.

whole datasets, compare the *FS* column in Table 4 and the *UAS* column in Table 1.

Given this observation, we proposed the following strategy for parsing subordinated clauses and we updated the *CCP* method as follows:

1. Find the longest sequence of neighboring subordinated clauses $C_1, C_2, \ldots, C_n$ so that $layer(C_{i+1}) = layer(C_i) + 1$ where $layer$ stands for a layer of embedding in a clause chart.

2. Create a sequence $S = C_1\ B_{1,2}\ C_2\ B_{2,3} \ldots C_n$ where $B_{i,i+1}$ is a boundary between C_i and C_{i+1}.

3. Using MST parse sequence S to get a dependency tree T_S.

Using the CCP strategy for parsing the `0B0` and `0B1` sentences, we can parse the `0B1B0` sentences so that we apply the CCP strategy for subordinated clauses first and subsequently for coordinated clauses. Table 5 presents the comparison of full-scale parsing and CCP.

Data	Sent.	FS	CCP
PDT dtest	166	81.72	82.98
PDT etest	160	81.98	84.22
CAC 2.0	885	79.68	80.84

Table 5: Parsing evaluation on the `0B1B0` sentences.

5.3 CCP as Full-scale Parsing

We have learned from the experiments that

1. it is efficient to parse coordinated clauses individually and connect their trees subsequently;

2. it is effective to parse a sequence of governing and dependent clauses at once.

Therefore we proposed and evaluated a final algorithm for dependency parsing that exploits sentence clause charts and a given dependency parser. The algorithm works in iterations. In each iteration, at least one layer of embedding in the clause chart is eliminated using the CCP strategy for `0B0` and `0B1` clauses.

Table 6 and Table 7 present the final comparison of full-scale parsing and the CCP strategy. The figures in Table 6 exclude simple sentences (one-clause sentences) from evaluation. We achieved an average 0.97% improvement in UAS when parsing all the sentences in the treebanks.

Data	Sent.	FS	CCP
PDT dtest	2,044	83.93	84.72
PDT etest	2,339	83.84	84.64
CAC 2.0	12,756	81.99	83.42

Table 6: Parsing evaluation on the sentences containing at least two clauses.

Data	Sent.	FS	CCP
PDT dtest	4,042	84.50	85.03
PDT etest	4,672	84.32	84.87
CAC 2.0	24,709	82.68	83.64

Table 7: Final comparison of full-scale parsing and CCP.

6 Future Work

Our motivation to address the task of parsing of long sentences arises from a project of extracting entities and relations from legal texts. We plan to apply the CCP strategy on the Czech Legal Text Treebank that contains significantly large number of long sentences than PDT 3.0. Consequently, we will do an eccentric evaluation of the RExtractor system to see whether better parsing results influence the extraction.

A sentence clause structure used in our experiments was generated automatically. However, the used procedure requires gold standard dependency trees on the input. We plan to develop an automatic procedure for obtaining the clause charts. This procedure will not require gold standard dependency trees on the input. Some experiments have been already done by Krůza and Kuboň (2009). In addition, we see several different approaches which could be implemented and evaluated.

In the presented experiments, we used the parser trained on complete sentences. However, the CCP strategy parses individual clauses in some situations. We believe that training a new model especially for clauses will bring a significant improvement. Another model could be trained for parsing sequences defined in Step 3 of proposed algorithm from Section 5.1.

Our parsing strategy is formulated to be language independent. The English part of the Czech-English Prague Dependency Treebank contains the entire Penn Treebank that is enhanced with segmentation of sentences into clauses.[4] We plan to apply the CCP strategy on this dataset.

7 Conclusion

In our pilot experiments, we showed that sentence clause charts improve dependency parsing of long sentences. We proposed a method that assumes segmentation of input sentences into clauses. Having such annotation at hand, we represent sentence clause structure using a clause chart that provides a layer of embedding for each clause in the sentence.

Our parsing strategy does not need to re-train a parser of one's choice. Instead of that, we separately parse coordinated and subordinated clauses with respect to the sentence clause chart and then connect their dependency trees.

The object language of our experiments is Czech and the parser used is a maximum spanning tree parser trained on the Prague Dependency Treebank. We achieved an average 0.97% improvement in the unlabeled attachment score.

Acknowledgments

We gratefully acknowledge support from the SVV project No. SVV 260 333 and from the LINDAT/CLARIN Research Infrastructure project of the Ministry of Education, Youth and Sports of the Czech Republic No. LM2015071. This work has also been using language resources developed and distributed by the LINDAT/CLARIN project. We thank three anonymous reviewers for their useful comments and remarks and we truly appreciate suggestions provided by Kemal Oflazer.

[4]`http://ufal.mff.cuni.cz/pcedt2.0/en/`

References

Steven Abney. 1996. Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, 2(04):337–344.

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. 2013. Prague Dependency Treebank 3.0. `http://ufal.mff.cuni.cz/pdt3.0`.

Fabio Ciravegna and Alberto Lavelli. 1999. Full Text Parsing Using Cascades of Rules: An Information Extraction Perspective. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pages 102–109, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ralph Debusmann, Denys Duchier, and Andreas Rossberg. 2005. Modular Grammar Design with Typed Parametric Principles. *Proceedings of FG-MOL 2005.*

Stefano Federici, Simonetta Montemagni, and Vito Pirrelli. 1996. Shallow Parsing and Text Chunking: a View on Underspecification in Syntax. *Cognitive Science Research Paper - University of Sussex CSRP*, pages 35–44.

Barbora Vidová Hladká, Jan Hajič, Jiří Hana, Jaroslava Hlaváčová, Jiří Mírovský, and Jan Raab. 2008. The Czech Academic Corpus 2.0 Guide. *The Prague Bulletin of Mathematical Linguistics*, (89):41–96.

Samar Husain, Phani Gadde, Joakim Nivre, and Rajeev Sangal. 2011. Clausal parsing helps data-driven dependency parsing: Experiments with Hindi. In *PProceedings of the 5th International Joint Conference on Natural Language Processing*, page 1279–1287, Chiang Mai, Thailand.

Vincent Kríž and Barbora Hladká. 2015. RExtractor: a Robust Information Extractor. In Matt Gerber, Catherine Havasi, and Finley Lacatusu, editors, *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 21–25, Denver, CO, USA. Association for Computational Linguistics.

Vincent Kríž, Barbora Hladká, Martin Nečaský, and Tomáš Knap. 2014. Data Extraction Using NLP Techniques and Its Transformation to Linked Data. In *Human-Inspired Computing and Its Applications - 13th Mexican International Conference on Artificial Intelligence, MICAI 2014, Tuxtla Gutiérrez, Mexico, November 16-22, 2014. Proceedings, Part I*, pages 113–124.

Oldřich Krůza and Vladislav Kuboň. 2009. Automatic Extraction of Clause Relationships from a

Treebank. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing. 10th International Conference, CICLing 2009, Mexico City, Mexico, March 1-7, 2009, Proceedings*, volume 5449 of *Lecture Notes in Computer Science*, pages 195–206, Berlin / Heidelberg. Springer.

Oldřich Krůza and Vladislav Kuboň. 2014. Automatic Recognition of Clauses. *International Journal of Computational Linguistics and Applications*, 5(1):125–138.

Vincent Kríž, Barbora Hladká, and Zdeňka Urešová. 2016. Czech Legal Text Treebank 1.0. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Vladislav Kuboň, Markéta Lopatková, Martin Plátek, and Patrice Pognan. 2007. A Linguistically-Based Segmentation of Complex Sentences. In David Wilson and Geoffrey Sutcliffe, editors, *Proceedings of FLAIRS 2007 (20th International Florida Artificial Intelligence Research Society Conference)*, pages 368–373, Key West, FL, USA. AAAI Press.

Vladislav Kuboň. 2001. *Problems of Robust Parsing of Czech*. Ph.D. thesis, Faculty of Mathematics and Physics, Charles University in Prague, Prague, Czech Republic.

Markéta Lopatková and Tomáš Holan. 2009. Segmentation Charts for Czech – Relations among Segments in Complex Sentences. In Adrian Dediu, Armand Ionescu, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications. Third International Conference, LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings*, volume 5457 of *Lecture Notes in Computer Science*, pages 542–553, Berlin / Heidelberg. Universitat Rovira i Virgili, Springer.

Markéta Lopatková, Petr Homola, and Natalia Klyueva. 2012. Annotation of Sentence Structure: Capturing the Relationship between Clauses in Czech Sentences. *Language Resources and Evaluation*, 46(1):25–36.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 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, BC, Canada. Association for Computational Linguistics.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An Effective Neural Network Model for Graph-based Dependency Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 313–322, Beijing, China, July. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, CoNLL'00, pages 127–132, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured Training for Neural Network Transition-Based Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China, July. Association for Computational Linguistics.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A Re-ranking Model for Dependency Parser with Recursive Convolutional Neural Network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1159–1168, Beijing, China, July. Association for Computational Linguistics.