An Empirical Investigation of Structured Output Modeling for Graph-based Neural Dependency Parsing

Zhisong Zhang, Xuezhe Ma, Eduard Hovy
Language Technologies Institute, Carnegie Mellon University
{zhisongz,xuezhem}@cs.cmu.edu, hovy@cmu.edu

Abstract

In this paper, we investigate the aspect of structured output modeling for the state-of-the-art graph-based neural dependency parser (Dozat and Manning, 2017). With evaluations on 14 treebanks, we empirically show that global output-structured models can generally obtain better performance, especially on the metric of sentence-level Complete Match. However, probably because neural models already learn good global views of the inputs, the improvement brought by structured output modeling is modest.

1 Introduction

In the past few years, dependency parsers, equipped with neural network models, have led to impressive empirical successes on parsing accuracy (Chen and Manning, 2014; Weiss et al., 2015; Dyer et al., 2015; Andor et al., 2016; Kiperwasser and Goldberg, 2016; Kuncoro et al., 2016; Dozat and Manning, 2017; Ma et al., 2018). Among them, the deep-biaffine attentional parser (**BiAF**) (Dozat and Manning, 2017) has stood out for its simplicity and effectiveness. BiAF adopts a simple bi-directional LSTM neural architecture (Ma and Hovy, 2016; Kiperwasser and Goldberg, 2016) with the first-order graph parsing algorithm (McDonald et al., 2005a,b). Simple as it appears to be, BiAF has led to several record-breaking performences in multiple treebanks and languages (Dozat et al., 2017).

In their pioneering work, besides the neural architecture, Dozat and Manning (2017) adopt a simple head-selection training object (Zhang et al., 2017) by regarding the original structured prediction task as an head-classification task in training. Although practically this simplification works well, there are still problems with it. Due to local normalization in the training objective (see §2.2), no global tree-structured information can be back-propagated during training. This can lead to the discrepancy between training and testing, since during testing, the MST (Maximum Spanning Tree) algorithm (McDonald et al., 2005b) is used to ensure valid tree structures. This problem raises concerns about the structured output layer. Several previous neural graph parsers utilized structured techniques (Pei et al., 2015; Kiperwasser and Goldberg, 2016; Zhang et al., 2016; Wang and Chang, 2016; Ma and Hovy, 2017), but their neural architectures might not be competitive to the current state-of-the-art BiAF parsing model. In this paper, building upon the BiAF based neural architecture, we empirically investigate the effectiveness of utilizing classical structured prediction techniques of output modeling for graph-based neural dependency parsing. We empirically show that structured output modeling can obtain better performance, especially on the the sentence-level metrics. However, the improvements are modest, probably because neural models make the problem easier to solve locally.

2 Output Modeling

In structured prediction tasks, a structured output $\mathbf{y}$ is predicted given an input $\mathbf{x}$. We refer to the encoding of the $\mathbf{x}$ as *input modeling*, and the modeling of the structured output $\mathbf{y}$ as *output modeling*.

Output modeling concerns modeling dependencies and interactions across multiple output components and assigning them proper scores. A common strategy to score the complex output structure is to factorize it into sub-structures, which is referred as *factorization*. A further step of *normalization* is needed to form the final score of an output structure. We will explain more details about these concepts in the situation of graph-based dependency parsing.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5592–5598
Florence, Italy, July 28 - August 2, 2019.

2.1 Factorization

The output structure of dependency parsing is a collection of dependency edges forming a single-rooted tree. Graph-based dependency parsers factorize the outputs into specifically-shaped sub-trees (factors). Based on the assumption that the sub-trees are independent to each other, the score of the output tree structure (T) is the combination of the scores of individual sub-trees in the tree.

In the simplest case, the sub-trees are the individual dependency edges connecting each modifier and its head word ((m, h)). This is referred to as first-order factorization (Eisner, 1996; McDonald et al., 2005a), which is adopted in (Dozat and Manning, 2017) and the neural parsing models in this work. There are further extensions to higher-order factors, considering more complex sub-trees with multiple edges (McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010; Ma and Zhao, 2012). We leave the exploration of these higher-order graph models to future work.

2.2 Normalization

After obtaining the individual scores of the sub-structures, we need to compute the score of the whole output structure. The main question is on what scale to normalize the output scores. For graph-based parsing, there can be mainly three options: Global, Local or Single, following different structured output constraints and corresponding to different loss functions.

Global Global models directly normalize at the level of overall tree structures, whose scores are obtained by directly summing the raw scores of the sub-trees without any local normalization. This can be shown clearly if further taking a probabilistic CRF-like treatment, where a final normalization is performed over all possible trees:

$$\text{Score}_g(T) = \log \frac{\exp \sum_{(m,h)\in T} \text{Score}(m, h)}{\sum_{T'} \exp \sum_{(m,h)\in T'} \text{Score}(m, h)}$$

Here, the normalization is carried out in the exact output space of all legal trees (T'). Max-Margin (Hinge) loss (Taskar et al., 2004) adopts the similar idea, though there is no explicit normalization in its formulation. The output space can be further constrained by requiring the projectivity of the trees (Kubler et al., 2009). Several manual-feature-based (McDonald et al., 2005b; Koo and Collins, 2010) and neural-based dependency parsers (Pei et al., 2015; Kiperwasser and Goldberg, 2016; Zhang et al., 2016; Ma and Hovy, 2017) utilize global normalization.

Local Local models, in contrast, ignore the global tree constraints and view the problem as a head-selection classification problem (Fonseca and Aluísio, 2015; Zhang et al., 2017; Dozat and Manning, 2017). The structured constraint that local models follow is that each word can be attached to one and only one head node. Based on this, the edge scores are locally normalized over all possible head nodes. This can be framed as the softmax output if taking a probabilistic treatment:

$$\text{Score}_l(T) = \sum_{(m,h)\in T} \log \frac{\exp \text{Score}(m, h)}{\sum_{h'} \exp \text{Score}(m, h')}$$

In this way, the model only sees and learns head-attaching decisions for each individual words. Therefore, the model is unaware of the global tree structures and may assign probabilities to non-tree cyclic structures, which are illegal outputs for dependency parsing. In spite of this defect, the local model enjoys its merits of simplicity and efficiency in training.

Single (Binary) If further removing the single-head constraint, we can arrive at a more simplified binary-classification model for each single edge, referred as the "Single" model, which predicts the presences and absences of dependency relation for every pair of words. Eisner (1996) first used this model in syntactic dependency parsing, and Dozat and Manning (2018) applied it to semantic dependency parsing. Here, the score of each edge is normalized against a fixed score of zero, forming a sigmoid output:

$$\text{Score}_s(T) = \sum_{(m,h)\in T} \log \frac{\exp \text{Score}(m, h)}{\exp \text{Score}(m, h) + 1}$$

Here, we only show the scoring formula for brevity. In training, since this binary classification problem can be quite imbalanced, we only sample partial of the negative instances (edges). Practically, we find a ratio of 2:1 makes a good balance, that is, for each token, we use its correct head word as the positive instance and randomly sample two other tokens in the sentence as negative instances.

2.3 Summary

The normalization methods that we describe above actually indicate the output structured constraints

Normalization	Loss	Algorithm
Single	Prob	–
Local	Prob	–
Global-NProj	Prob Hinge	Matrix-Tree Theorem Chu-Liu-Edmonds
Global-Proj	Prob Hinge	Inside-Outside Eisner's

Table 1: Summarization of the methods explored in this work and their corresponding algorithms.

that the model is aware of. The global model is aware of all the constraints to ensure a legal dependency tree. The local model maintains the single-head constraint while there are almost no structured constrains under the single model. To be noted, for all these normalization methods, we can take various loss functions. In this work, we study two typical ones: probabilistic Maximum-Likelihood loss (Prob), which requires actual normalization over the output space, and Max-Margin Hinge loss (Hinge), which only requires loss-augmented decoding in the same output space.

Table 1 summarizes the methods (normalization and loss function) that we investigate in our experiments. For global models, we consider both Projective (Proj) and Non-Projective (NProj) constraints. Specific algorithms are required for probabilistic loss (a variation of Inside-Outside algorithm for projective (Paskin, 2001) and Matrix-Tree Theorem for non-projective parsing (Koo et al., 2007; Smith and Smith, 2007; McDonald and Satta, 2007)) and hinge loss (Eisner's algorithm for projective (Eisner, 1996) and Chu-Liu-Edmonds' algorithm for non-projective parsing (Chu and Liu, 1965; Edmonds, 1967; McDonald et al., 2005b)). For Single and Local models, we only utilize probabilistic loss, since in preliminary experiments we found hinge loss performed worse. No special algorithms other than simple enumeration are needed for them in training. In testing, we adopt non-projective algorithms for the non-global models unless otherwise noted.

3 Experiments

3.1 Settings

We evaluate the parsers on 14 treebanks: English Penn Treebank (PTB), Penn Chinese Treebank (CTB) and 12 selected treebanks from Universal Dependencies (v2.3) (Nivre et al., 2018). We follow standard data preparing conventions as in Ma et al. (2018). Please refer to the supplementary material for more details of data preparation.

For the neural architecture, we also follow the settings in Dozat and Manning (2017) and Ma et al. (2018) and utilize the deep BiAF model. For the input, we concatenate representations of word, part-of-speech (POS) tags and characters. Word embeddings are initialized with the pre-trained fasttext word vectors[1] for all languages. For POS tags and Character information, we use POS embeddings and a character-level Convolutional Neural Network (CNN) for the encoding. For the encoder, we adopt three layers of bi-directional LSTM to get contextualized representations, while our decoder is the deep BiAF scorer as in Dozat and Manning (2017). We only slightly tune hyper-parameters on the Local model and the development set of PTB, and then use the same ones for all the models and datasets. More details of hyper-parameter settings are provided in the supplementary material. Note that our exploration only concerns the final output layer which does not contain any trainable parameters in the neural model, and all our comparisons are based on exactly the same neural architecture and hyper-parameter settings. Only the output normalization methods and the loss functions are different.

We run all the experiments with our own implementation[2], which is written with PyTorch. All experiments are run with one TITAN-X GPU. In training, global models take around twice the time of the local and single models; while in testing, their decoding costs are similar.

3.2 Results

We run all the models three times with different random initialization, and the averaged results on the test sets are shown in Table 2. Due to space limitation, we only report LAS (Labeled Attachment Score) and LCM (Labeled Complete Match) in the main content. We also include the unlabeled scores UAS (Unlabeled Attachment Score) and UCM (Unlabeled Complete Match) in the supplementary material. The evaluations on PTB and CTB exclude punctuations[3], while on UD we evaluate on all tokens (including punctuations) as the setting of the LAS metric in the CoNLL shared tasks (Zeman et al., 2017, 2018).

[1] https://fasttext.cc/docs/en/pretrained-vectors.html

[2] Our implementation is publicly available at https://github.com/zzsfornlp/zmsp

[3] Tokens whose gold POS tag is one of {“ ” : , .} for PTB or “PU” for CTB

Method	Single	Local	Global-NProj		Global-Proj	
	Prob	Prob	Prob	Hinge	Prob	Hinge
PTB	93.43/44.67	93.75/46.65	93.84†/47.17	93.91†/47.78†	93.79/47.16	**93.96**†/**48.47**†
CTB	87.03/31.26	88.16/33.16	88.26/33.73	87.92/32.77	**88.46**†/**35.11**†	88.14/34.00†
bg-btb	89.97/39.25	90.06/39.99	90.35†/**41.25**†	**90.42**†/40.83	90.15/40.98	90.20/40.53
ca-ancora	91.23/25.03	91.54/26.35	91.73†/27.19†	**91.73**†/26.65	91.39/**27.39**†	91.51/27.19†
cs-pdt	90.95/43.07	91.51/45.62	**91.69**†/**46.60**†	91.52/46.02†	91.10/44.43	91.18/44.02
de-gsd	**83.68**†/22.65	83.43/22.42	83.65†/22.86	83.66†/22.93	83.39/23.37†	83.63/**23.51**†
en-ewt	88.01/55.93	88.33/56.46	88.52†/57.29†	**88.59**†/57.33†	88.52†/**58.29**†	88.41/57.31†
es-ancora	90.82/27.27	91.05/27.41	91.12/27.89	**91.14**/27.35	90.84/**28.41**†	91.03/27.70
fr-gsd	88.00/20.03	88.13/20.83	88.43†/21.71	88.22/20.27	**88.59**†/**23.80**†	88.41†/21.88
it-isdt	91.71/44.05	92.01/44.26	92.16/45.30	92.08/45.02	**92.49**†/**48.27**†	92.37†/46.75†
nl-alpino	88.31/33.11	88.81/33.67	**88.94**/34.62	88.94/**35.12**†	88.37/33.05	88.45/33.00
no-bokmaal	92.89/53.60	92.89/53.58	**93.02**†/**54.36**†	92.78/53.09	92.82/53.57	92.70/52.71
ro-rrt	85.10†/12.85†	84.58/11.57	84.85†/12.44	85.04†/13.03†	84.89†/12.94†	**85.16**†/**13.76**†
ru-syntagrus	92.76/48.67	93.29/50.69	**93.36**†/**50.97**	93.29/50.72	93.11/50.79	93.19/50.17
Average	89.56/35.82	89.82/36.62	**89.99**†/37.39†	89.95†/37.07†	89.85/**37.68**†	89.88/37.21†

Table 2: Results (**LAS/LCM**) on the test sets (averaged over three runs). '†' means that the result of the model is statistically significantly better (by permutation test, $p < 0.05$) than the Local-Prob model.

Overall, the global models[4] perform better consistently, especially on the metrics of Complete Match, showing the effectiveness of being aware of global structures. However, the performance gaps between global models and local models are small. More surprisingly, the single models that ignore all the structures only lag behind by around 0.4 averagely. In some way, this shows that input modeling, including the distributed input representations, contextual encoders and parts of the decoders, makes the structured decision problem easier to solve locally. Neural models seem to squeeze the improvement space that structured output modeling can bring.

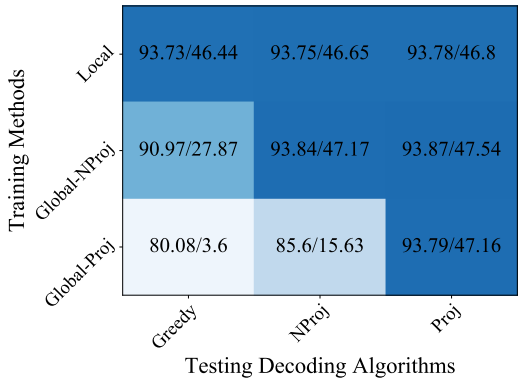

Figure 1: Results (**LAS/LCM**, on the PTB test set) of different models (with prob loss) and decoding algorithms. Rows represent the methods used in training and columns denote the decoding algorithms in testing. Darker colors represent better scores.

3.3 Analysis

We further analyze on output constraints and input modeling. For brevity, we only analyze on PTB and use probabilistic models. Single models are excluded for their poorer performance.

Firstly, we study the influence of output constraint differences in training and testing. Here, we include a naive "Greedy" decoding algorithm which simply selects the most probable head for each token. This does not ensure that the outputs are trees and corresponds to the head-classification method adopted by local models. The results of different models and training/testing algorithms are shown in Figure 1. Interestingly, the discrepancies in training and testing are only detrimental when the output constraint in testing is looser than that in training (the left corner in the figure), as shown by the poorer results in the training-testing pairs of "NProj-Greedy", "Proj-Greedy" and "Proj-NProj". Generally, projective decoding is the best choice since PTB contains mostly (99.9%) projective trees.

Next, we study the interactions of "weaker" neural architectures (for input modeling) and output modeling. We consider three "weaker" models: (1) "No-Word" ignores all the lexical inputs and is a pure delexicalized model; (2) "Simple-CNN" replaces the RNN encoder with a much simpler encoder, which is a simple single-layer CNN with a window size of three for the purpose of studying weak models; (3) "No-Encoder" com-

[4]Projective global models perform averagely poorer than non-projective ones, since some of the treebanks (for example, only 88% of the trees in 'cs-pdt' are projective) contain a non-negligible portion of non-projective trees.

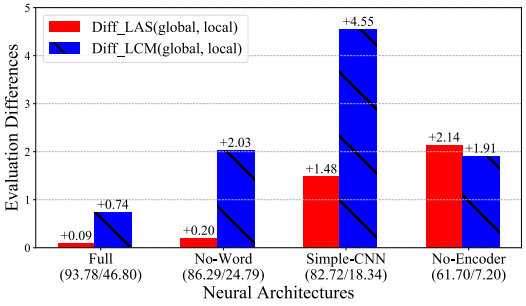

Figure 2: Evaluation differences (on the PTB test set) between global and local methods when adopting various "weaker" neural architectures. Numbers below x-axis labels denote the evaluation scores (LAS/LCM) of the local models.

pletely deletes the encoder, leading to a model that does not take any contextual information. Here, since we are testing on PTB which almost contain only projective trees, we use projective decoding for all models. As shown in Figure 2, when input modeling is weaker, the improvements brought by the global model generally get larger. Here, the LCM for "No-Encoder" is an outlier, probably because this model is too weak to get reasonable complete matches. The results show that with weaker input modeling, the parser can generally benefit more from structured output modeling. In some way, this also indicates that better input modeling can make the problem depend less on the global structures so that local models are able to obtain competitive performance.

4 Discussion and Conclusion

In this paper, we call the models that are aware of the whole output structures "global". In fact, with the neural architecture that can capture features from the whole input sentence, actually all the models we explore have a "global" view of inputs. Our experiments show that with this kind of global input modeling, good results can be obtained even when ignoring certain output structures, and further enhancement of global output structures only provides small benefits. This might suggest that input and output modeling can capture certain similar information and have overlapped functionalities for the structured decisions.

In future work, there can be various possible extensions. We will explore more about the interactions between input and output modeling for structured prediction tasks. It will be also interesting to adopt even stronger input models, especially, those enhanced with contextualized representations from Elmo (Peters et al., 2018) or BERT (Devlin et al., 2018). A limitation of this work is that we only explore first-order graph based parser, that is, for the *factorization* part, we do not consider high-order sub-subtree structures. This part will surely be interesting and important to explore.

Acknowledgement

This research was supported in part by DARPA grant FA8750-18-2-0018 funded under the AIDA program.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, Prague, Czech Republic. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings*

of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 20–30.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. 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 334–343, Beijing, China. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards, B*, 71:233–240.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345, Copenhagen.

Erick Fonseca and Sandra Aluísio. 2015. A deep architecture for non-projective dependency parsing. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 56–61, Denver, Colorado. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden. Association for Computational Linguistics.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic. Association for Computational Linguistics.

Sandra Kubler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan & Claypool Publishers.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1744–1753, Austin, Texas. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective mst parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 59–69, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Xuezhe Ma and Hai Zhao. 2012. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 785–796, Mumbai, India. The COLING 2012 Organizing Committee.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan. Association for Computational Linguistics.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the ACL (EACL 2006)*, pages 81–88. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. 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. Association for Computational Linguistics.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 121–132, Prague, Czech Republic. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, and et al. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Mark A Paskin. 2001. Cubic-time parsing and learning algorithms for grammatical bigram models. Technical report.

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. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 132–140, Prague, Czech Republic. Association for Computational Linguistics.

Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proceedings of EMNLP 2004*, pages 1–8, Barcelona, Spain. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315, Berlin, Germany. 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. Association for Computational Linguistics.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571. Association for Computational Linguistics.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1382–1392, Berlin, Germany. Association for Computational Linguistics.