Second-Order Neural Dependency Parsing with Message Passing and End-to-End Training

Xinyu Wang and **Kewei Tu**[*]
School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
{wangxy1,tukw}@shanghaitech.edu.cn

Abstract

In this paper, we propose second-order graph-based neural dependency parsing using message passing and end-to-end neural networks. We empirically show that our approaches match the accuracy of very recent state-of-the-art second-order graph-based neural dependency parsers and have significantly faster speed in both training and testing. We also empirically show the advantage of second-order parsing over first-order parsing and observe that the usefulness of the head-selection structured constraint vanishes when using BERT embedding.

1 Introduction

Graph-based dependency parsing is a popular approach to dependency parsing that scores parse components of a sentence and then finds the highest scoring tree through inference. First-order graph-based dependency parsing takes individual dependency edges as the components of a parse tree, while higher-order dependency parsing considers more complex components consisting of multiple edges. There exist both exact inference algorithms (Carreras, 2007; Koo and Collins, 2010; Ma and Zhao, 2012) and approximate inference algorithms (McDonald and Pereira, 2006; Smith and Eisner, 2008; Gormley et al., 2015) to find the best parse tree. Recent work focused on neural network based graph dependency parsers (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016; Cheng et al., 2016; Kuncoro et al., 2016; Ma and Hovy, 2017; Dozat and Manning, 2017). Dozat and Manning (2017) proposed a first-order graph-based neural dependency parsing approach with a simple head-selection training objective. It uses a biaffine function to score dependency edges and has high efficiency and good performance. Subsequent work introduced second-order inference into their parser. Ji et al. (2019) proposed a graph neural network that captures second-order information in token representations, which are then used for first-order parsing. Very recently, Zhang et al. (2020) proposed an efficient second-order tree CRF model for dependency parsing and achieved state-of-the-art performance.

In this paper, we first show how a previously proposed second-order semantic dependency parser (Wang et al., 2019) can be applied to syntactic dependency parsing with simple modifications. The parser is an end-to-end neural network derived from message passing inference on a conditional random field that encodes the second-order parsing problem. We then propose an alternative conditional random field that incorporates the head-selection constraint of syntactic dependency parsing, and derive a novel second-order dependency parser. We empirically compare the two second-order approaches and the first-order baselines on English Penn Tree Bank 3.0 (PTB), Chinese Penn Tree Bank 5.1 (CTB) and datasets of 12 languages in Universal Dependencies (UD). We show that our approaches achieve state-of-the-art performance on both PTB and CTB and our approaches are significantly faster than recently proposed second-order parsers.

We also make two interesting observations from our empirical study. First, it is a common belief that contextual word embeddings such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) already conveys sufficient high-order information that renders high-order parsing less useful, but we find that second-order decoding is still helpful even with strong contextual embeddings like BERT. Second, while Zhang et al. (2019) previously found that incoperating the head-selection constraint is helpful in first-order parsing, we find that with a better loss function design and hyper-parameter tun-

[*] Kewei Tu is the corresponding author.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 93–99
December 4 - 7, 2020.

ing both first- and second-order parsers without the head-selection constraint can match the accuracy of parsers with the head-selection constraint and can even outperform the latter when using BERT embedding.

Our approaches are closely related to the work of Gormley et al. (2015), which proposed a non-neural second-order parser based on Loopy Belief Propagation (LBP). Our work differs from theirs in that: 1) we use Mean Field Variational Inference (MFVI) instead of LBP, which Wang et al. (2019) found is faster and equally accurate in practice; 2) we add the head-selection constraint and do not include the global tree constraint that is shown to produce only slight improvement (Zhang et al., 2019) but would complicate our neural network design and implementation; 3) we employ modern neural encoders and achieve much better parsing accuracy. Our approaches are also closely related to the very recent work of Fonseca and Martins (2020). The main difference is that we use MFVI while they use the dual decomposition algorithm AD^3 (Martins et al., 2011, 2013) for approximate inference.

2 Approach

Zhang et al. (2019) categorized different kinds of graph-based dependency parsers based on their structured output constraints according to the normalization for output scores. A **Local** approach views dependency parsing as a head-selection problem, in which each word selects exactly one dependency head. A **Single** approach places no structured constraint, viewing the existence of each possible dependency edge as an independent binary classification problem.

The second-order semantic dependency parser of Wang et al. (2019) is an end-to-end neural network derived from message passing inference on a conditional random field that encodes the second-order parsing problem. It is clearly a **Single** approach because of the lack of structured constraints in semantic dependency parsing. We can apply this approach to syntactic dependency parsing with two minor modifications. First, co-parents, one of the three types of second-order parts, become invalid and hence are removed. Second, for the approach to output valid parse trees during testing, we run maximum spanning tree (MST) (McDonald et al., 2005) based on the posterior edge probabilities predicted by the approach.

Inspired by Wang et al. (2019), below we propose a **Local** second-order parsing approach. While the **Single** approach uses Boolean random variables to represent existence of possible dependency edges, our **Local** approach defines a discrete random variable for each word specifying its dependency head, thus enforcing the head-selection constraint and leading to different formulation of the message passing inference steps.

2.1 Scoring

Following Dozat and Manning (2017), we predict edge existence and edge labels separately. Suppose the input sentence is $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$ where w_0 is a dummy root. We feed word representations outputted by the BiLSTM encoder into a biaffine function to assign score $s_{ij}^{(\text{edge})}$ to edge $w_i \rightarrow w_j$. We use a Trilinear function to assign score $s_{ij,ik}^{(\text{sib})}$ to the siblings part consisting of edges $w_i \rightarrow w_j$ and $w_i \rightarrow w_k$, and another Trilinear function to assign score $s_{ij,jk}^{(\text{gp})}$ to the grandparent part consisting of edges $w_i \rightarrow w_j$ and $w_j \rightarrow w_k$. For edge labels, we use a biaffine function to predict label scores of each potential edge and use a softmax function to compute the label distribution $P(y_{ij}^{(\text{label})}|\mathbf{w})$, where $y_{ij}^{(\text{label})}$ represents the possible label for edge $w_i \rightarrow w_j$.

2.2 Message Passing

The head-selection structured constraint requires that each word except the root has exactly one head. We define variable $X_j \in \{0, 1, 2, \ldots, n\}$ to indicate the head of word w_j. We then define a conditional random field (CRF) over $[X_1, \ldots, X_n]$. For each variable X_j, the unary potential is defined by:

$$\phi_u(X_j = i) = \exp(s_{ij}^{(\text{edge})})$$

Given two variables X_j and X_l, the binary potential is defined by:

$$\phi_p(X_j = i, X_l = k) = \begin{cases} \exp(s_{ij,kl}^{(\text{sib})}) & k = i \\ \exp(s_{ij,kl}^{(\text{gp})}) & k = j \\ 1 & \text{Otherwise} \end{cases}$$

We use MFVI for approximate inference on this CRF. The algorithm updates the factorized posterior distribution $Q_j(X_j)$ of each word iteratively.

$$\begin{aligned}\mathcal{M}_j^{(t-1)}(i) = &\sum_{k \neq i,j} Q_k^{(t-1)}(i) s_{ij,ik}^{(sib)} \\ &+ Q_k^{(t-1)}(j) s_{ij,jk}^{(gp)} + Q_i^{(t-1)}(k) s_{ki,ij}^{(gp)} \\ Q_j^{(t)}(i) = &\frac{\exp\{s_{ij}^{(\text{edge})} + \mathcal{M}_j^{(t-1)}(i)\}}{\sum_{k=0}^{n} \exp\{s_{kj}^{(\text{edge})} + \mathcal{M}_j^{(t-1)}(k)\}}\end{aligned}$$

At $t = 0$, $Q_j^{(t)}(X_j)$ is initialized by normalizing the unary potential. The iterative update steps can be unfolded as recurrent neural network layers parameterized by part scores, thus forming an end-to-end neural network.

Compared with the update formula in the **Single** approach, here the posterior distributions are defined over head-selections and are normalized over all possible heads. The computational complexity remains the same.

2.3 Learning

We define the cross entropy losses by:

$$\begin{aligned}\mathcal{L}^{(\text{edge})} &= -\sum_{i} \log[Q_i(y_i^{*(\text{edge})}|\mathbf{w})] \\ \mathcal{L}^{(\text{label})} &= -\sum_{i,j} \mathbb{1}(y_j^{*(\text{edge})} = i) \log(P(y_{ij}^{*(\text{label})}|\mathbf{w})) \\ \mathcal{L} &= \lambda \mathcal{L}^{(\text{label})} + (1 - \lambda)\mathcal{L}^{(\text{edge})}\end{aligned}$$

where $y_i^{*(\text{edge})}$ is the head of word w_i and $y_{ij}^{*(\text{label})}$ is the label of edge $w_i \rightarrow w_j$ in the golden parse tree, λ is a hyper-parameter and $\mathbb{1}(x)$ is an indicator function that returns 1 when x is true and 0 otherwise.

3 Experiments

3.1 Setups

Following previous work (Dozat and Manning, 2017; Ma et al., 2018), we use PTB 3.0 (Marcus et al., 1993), CTB 5.1 (Xue et al., 2002) and 12 languages in Universal Dependencies (Nivre et al., 2018) (UD) 2.2 to evaluate our parser. Punctuation is ignored in all the evaluations. We use the same treebanks and preprocessing as Ma et al. (2018) for PTB, CTB, and UD. For all the datasets, we remove sentences longer than 90 words in training sets for faster computation.

Hidden Layer	**Hidden Sizes**
Word/GloVe/Char	100
POS	50
GloVe Linear	125
BERT Linear	125
BiLSTM	3*600
Char LSTM	1*400
Unary Arc (UD)	500
Local1O/Local2O Unary Arc (Others)	450
Single1O/Single2O Unary Arc (Others)	550
Label	150
Binary Arc	150
Dropouts	**Dropout Prob.**
Word/GloVe/POS	20%
Char LSTM (FF/recur)	33%
Char Linear	33%
BiLSTM (FF/recur)	45%/25%
Unary Arc/Label	25%/33%
Binary Arc	25%
Optimizer & Loss	**Value**
Local1O/Local2O Interpolation (λ)	0.40
Single1O/Single2O Interpolation (λ)	0.07
Adam β_1	0
Adam β_2	0.95
Decay Rate	0.85
Decay Step (without **dev** improvement)	500
Weight Initialization	**Mean/Stddev**
Unary weight	0.0/1.0
Binary weight	0.0/0.25

Table 1: Hyper-parameter for **Local1O**, **Single2O** and **Local2O** in our experiment.

We use **GNN**, **Local1O**, **Single1O**, **Local2O** and **Single2O** to represent the approaches of Ji et al. (2019), Dozat and Manning (2017), Dozat and Manning (2018), and our two second-order approaches respectively. For all the approaches, we use the MST algorithm to guarantee tree-structured output in testing. We use the concatenation of word embeddings, character-level embeddings and part-of-speech (POS) tag embeddings to represent words and additionally concatenate BERT embeddings for experiments with BERT. For a fair comparison with previous work, we use GloVe (Pennington et al., 2014) and BERT-Large-Uncased model for PTB, and structured-skipgram (Ling et al., 2015) and BERT-Base-Chinese model for CTB. For UD, we use fastText embeddings (Bojanowski et al., 2017) and BERT-Base-Multilingual-Cased model for different languages. We set the default iteration number for our approaches to 3 because we find no improvement on more or less iterations.

For **GNN**[1], we rerun the code based on the official release of Ji et al. (2019). For **Single1O**, **Local1O**[2], **Single2O**[3], we implement these ap-

[1] https://github.com/AntNLP/gnn-dep-parsing

[2] https://github.com/tdozat/Parser-v3

[3] https://github.com/wangxinyu0922/Second_Order_SDP

	PTB		CTB	
	UAS	LAS	UAS	LAS
Dozat and Manning (2017)	95.74	94.08	89.30	88.23
Ma et al. (2018)♠	95.87	94.19	90.59	89.29
F&G (2019)♠	96.04	94.43	-	-
GNN	95.87	94.15	90.78	89.50
Single1O	95.75	94.04	90.53	89.28
Local1O	95.83	94.23	90.59	89.28
Single2O	95.86	94.19	90.75	89.55
Local2O	95.98	94.34	**90.81**	**89.57**
Ji et al. (2019)†	95.97	94.31	-	-
Zhang et al. (2020)†‡	**96.14**	**94.49**	-	-
Local2O†‡	96.12	94.47	-	-
+BERT				
Zhou and Zhao (2019)♣	97.20	95.72		
Clark et al. (2018)◇	96.60	95.00	-	-
Single1O	96.82	95.20	92.73	91.64
Local1O	96.86	95.32	92.47	91.30
Single2O	96.86	95.31	**92.78**	**91.69**
Local2O	**96.91**	**95.34**	92.55	91.38

Table 2: Comparison of our approaches and the previous state-of-the-art approaches on PTB and CTB. We report our results averaged over 5 runs. †: These approaches perform model selection based on the score on the development set. ‡: These approaches do not use POS tags as input. ◇: Clark et al. (2018) uses semi-supervised multi-task learning with ELMo embeddings. ♠: These approaches use structured-skipgram embeddings instead of GloVe embeddings for PTB. ♣: For reference, Zhou and Zhao (2019) utilized both dependency and constituency information in their approach. Therefore, the results are not comparable to our results.

proaches based on the official release code of Wang et al. (2019) and we implement **Local2O** based on this code. In speed comparison, we implement the second-order approaches based on an PyTorch implementation biaffine parser[4] implemented by Zhang et al. (2020) for a fair speed comparison with their approach[5]. Since we find that the accuracy of our approaches based on PyTorch implementation on PTB does not change, we only report scores based on Wang et al. (2019).

3.2 Hyper-parameters

The hyper-parameters we used in our experiments is shown in Table 1. We tune the the hidden size for calculating $s_{ij}^{(\text{edge})}$ (Unary Arc in the table) separately for PTB and CTB. Following Qi et al. (2018), we switch to AMSGrad (Reddi et al., 2018) after 5,000 iterations without improvement. We train models for 75,000 iterations with batch sizes of 6000 tokens and stopped the training early after 10,000 iterations without improvements on development sets. Different from previous approaches such as Dozat and Manning (2017) and Ji et al. (2019), we use Adam (Kingma and Ba, 2015) with a learning rate of 0.01 and anneal the learning rate by 0.85 for every 500 iterations without improvement on the development set for optimization. For **GNN**, we train the models with the same setting as in Ji et al. (2019). We do not use character embeddings and our optimization settings for **GNN** because we find they do not improve the accuracy.

For the edge loss of **Single** approaches, Zhang et al. (2019) proposed to sample a subset of the negative edges to balance positive and negative examples, but we find that using a relatively small interpolation λ (shown in Table 1) on label loss can improve the accuracy and the sampling does not help further improve the accuracy.

3.3 Results

Table 2 shows the Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS) of all the approaches as well as the reported scores of previous state-of-the-art approaches on PTB and CTB. It can be seen that without BERT, our **Local2O** achieves state-of-the-art performance on CTB and has almost the same accuracy as the very recent work of Zhang et al. (2020) on PTB. With BERT embeddings, **Local2O** performs the best on PTB while **Single2O** has the best accuracy on CTB.

Table 3 shows the results of the five approaches on UD in addition to PTB and CTB. We make the following observations. First, our second-order approaches outperform **GNN** and the first-order approaches both with and without BERT embeddings, showing that second-order decoders are still helpful in neural parsing even with strong contextual embeddings. Second, without BERT, **Local** slightly outperforms **Single**, although the difference between the two is quite small[6]; when BERT is used, however, **Single** clearly outperforms **Local**, which is quite interesting and warrants further investigation in the future. Third, the relative strength of **Local** and **Single** approaches varies over treebanks, suggesting varying importance of the head-selection constraint.

[4] `https://github.com/yzhangcs/parser`

[5] At the time we finished the paper, the official code for the second-order tree CRF parser have not release yet. We believe it is a fair comparison since we use the same settings and GPU as Zhang et al. (2020).

[6] Note that Zhang et al. (2019) reports higher difference in accuracy between first-order **Local** and **Single** approaches. The discrepancy is most likely caused by our better designed loss function and tuned hyper-parameters.

	PTB	**CTB**	**bg**	**ca**	**cs**	**de**	**en**	**es**	**fr**	**it**	**nl**	**no**	**ro**	**ru**	**Avg.**
GNN	94.15	89.50†	90.33	92.39	90.95	79.73	88.43	91.56	87.23	92.44	88.57	89.38	85.26	91.20	89.37
Single1O	94.04	89.28	90.05	92.72†	92.07	81.73	89.55	92.10	88.27	92.64	89.57	91.81	85.39	92.60	90.13
Local1O	94.23	89.28	90.30	92.56	**92.15**	81.42	89.43	91.99	88.26	92.49	89.76	**91.91**	85.27	**92.72**	90.13
Single2O	94.19	89.55†	90.24	92.82†	92.13	**81.99**†	89.64†	**92.17**†	**88.69**	**92.83**†	89.97†	91.90	85.53†	92.58	90.30†
Local2O	**94.34**†‡	**89.57**†	**90.53**†	**92.83**†	92.12	81.73	**89.72**†	92.07	88.53	92.78	**90.19**†	91.88	**85.88**†‡	92.67	**90.35**†
+BERT															
Single1O	95.20	91.64†	90.87	93.55†	92.01	81.95†	90.44†	92.56†	**89.35**	93.44†	90.89	91.78	86.13†	92.51	90.88†
Local1O	95.32	91.30	91.03	93.17	91.93	81.66	90.09	92.32	89.26	93.05	90.93	91.62	85.67	92.51	90.70
Single2O	95.31	**91.69**†‡	**91.30**†	**93.60**†‡	**92.09**†	**82.00**†‡	**90.75**†‡	**92.62**†‡	89.32	**93.66**†	**91.21**	**91.74**	**86.40**†	92.61	**91.02**†‡
Local2O	**95.34**	91.38	91.13	93.34†	92.07†	81.67	90.43†	92.45†	89.26	93.50†	90.99	91.66	86.09†	**92.66**	90.86†

Table 3: LAS and standard deviations on test sets. We report results averaged over 5 runs. We use ISO 639-1 codes to represent languages from UD. † means that the model is statistically significantly better than the **Local1O** model by Wilcoxon rank-sum test with a significance level of $p < 0.05$. We use ‡ to represent winner of the significant test between the **Single2O** and **Local2O** models.

System	Train	Test	Time Complexity
GNN	392	464	$O(n^2d)$
Zhang et al. (2020)	200	400	$O(n^3)$
Single1O	616	1123	$O(n^2)$
Local1O	**625**	**1150**	$O(n^2)$
Single2O	481	966	$O(n^3)$
Local2O	486	1006	$O(n^3)$

Table 4: Comparison of training and testing speed (sentences per second) and the time complexity of the decoders of different approaches on PTB.

3.4 Speed Comparison

We evaluate the speed of different approaches on a single GeForce GTX 1080 Ti GPU following the setting of Zhang et al. (2020). As shown in Table 4, our **Local** approach and **Single** approach have almost the same speed. Our second-order approaches only slow down the training and testing speed in comparison with the first-order approaches by 23% and 12% respectively. They are also significantly faster than previous state-of-the-art approaches. Our **Local** approach is 1.2 and 2.3 times faster than **GNN** in training and testing respectively and is 2.4 and 2.9 times faster than the second-order tree CRF approach of Zhang et al. (2020).

In terms of time complexity, our second-order decoders have a time complexity of $O(n^3)$[7]; while the time complexity of **GNN** is $O(n^2d)$, the hidden size d (500 by default) is typically much larger than sentence length n; and the decoder of Zhang et al. (2020) has a time complexity of $O(n^3)$ as well, but it requires sequential computation over the input sentence while our decoders can be parallelized over words of the input sentence.

4 Conclusion

We propose second-order graph-based dependency parsing based on message passing and end-to-end neural networks. We modify a previous approach that predicts dependency edges independently and also design a new approach that incorporates the head-selection structured constraint. Our experiments show that our second-order approaches have better overall performance than the first-order baselines; they achieve competitive accuracy with very recent start-of-the-art second-order graph-based parsers and are significantly faster. Our empirical comparisons also show that second-order decoders still outperform first-order decoders even with BERT embeddings, and that the usefulness of the head-selection constraint is limited, especially when using BERT embeddings. Our code is publicly avilable at `https://github.com/wangxinyu0922/Second_Order_Parsing`.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61976139).

References

Joakim Nivre et al. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[7] The MST algorithm has a time complexity of $O(n^2)$ and we follow Dozat et al. (2017) only using the MST algorithm when the argmax predictions of structured output are not trees.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2204–2214, Austin, Texas. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

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. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Left-to-right dependency parsing with pointer networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 710–716, Minneapolis, Minnesota. Association for Computational Linguistics.

Erick Fonseca and André F. T. Martins. 2020. Revisiting higher-order dependency parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Matthew R. Gormley, Mark Dredze, and Jason Eisner. 2015. Approximation-aware dependency parsing by belief propagation. *Transactions of the Association for Computational Linguistics*, 3:489–501.

Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based dependency parsing with graph neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, Florence, Italy. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

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.

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.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of Word2Vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado. 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.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

André Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria. Association for Computational Linguistics.

André Martins, Noah Smith, Mário Figueiredo, and Pedro Aguiar. 2011. Dual decomposition with many overlapping components. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 238–249, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

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, British Columbia, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *International Conference on Learning Representations*.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 145–156, Honolulu, Hawaii. 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.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618, Florence, Italy. Association for Computational Linguistics.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated Chinese corpus. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order treecrf for neural dependency parsing. *arXiv preprint arXiv:2005.00975*.

Zhisong Zhang, Xuezhe Ma, and Eduard Hovy. 2019. An empirical investigation of structured output modeling for graph-based neural dependency parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5592–5598, Florence, Italy. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.