Improved Dependency Parsing using Implicit Word Connections Learned from Unlabeled Data

Wenhui Wang[†] **Baobao Chang**[†] **Mairgup Mansur**[‡]
[†]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[‡]Sogou Technology Inc., Beijing, China
{wenwan,chbb}@pku.edu.cn
{maerhufu}@sogou-inc.com

Abstract

Pre-trained word embeddings and language model have been shown useful in a lot of tasks. However, both of them cannot directly capture word connections in a sentence, which is important for dependency parsing given its goal is to establish dependency relations between words. In this paper, we propose to implicitly capture word connections from unlabeled data by a word ordering model with self-attention mechanism. Experiments show that these implicit word connections do improve our parsing model. Furthermore, by combining with a pre-trained language model, our model gets state-of-the-art performance on the English PTB dataset, achieving 96.35% UAS and 95.25% LAS.

1 Introduction

Dependency parsing is a fundamental task for language processing which aims to establish syntactic relations between words in a sentence. Graph-based models (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010) and transition-based models (Nivre, 2008; Zhang and Nivre, 2011) are the most successful solutions to the challenge.

Recently, neural network methods have been successfully introduced into dependency parsing. Deep feed-forward neural network models (Chen and Manning, 2014; Pei et al., 2015; Weiss et al., 2015) are proposed firstly. It alleviates the heavy burden of feature engineering. LSTM networks (Hochreiter and Schmidhuber, 1997) are then applied to dependency parsing (Dyer et al., 2015; Cross and Huang, 2016; Wang and Chang, 2016; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016) due to its ability to capture contextual information. Generative neural network models (Dyer et al., 2016; Smith et al., 2017; Choe and Charniak, 2016) also show promising parsing performance.

Different from Machine Translation task where massive sets of labeled data could be easily obtained, parsing performance is limited by the relatively small size of available treebank. Vinyals et al. (2015) and Weiss et al. (2015) adopt an approach of tri-training to augment the labeled data. They generate large quantities of parse trees by parsing unlabeled data with two existing parsers and selecting only the sentences for which the two parsers produced the same trees. However, the trees produced this way have noise[1] and tend to be short sentences, since it is easier for different parsers to get consistent results.

Pre-trained neural networks are another methods to take advantage of unlabeled data. Pre-trained word embeddings (Mikolov et al., 2013) and language model (Józefowicz et al., 2016; Peters et al., 2017, 2018) have been shown useful in modelling NLP tasks since word embeddings could capture word semantic information and language model could capture contextual information at the sentence level. However, connections between words in the sentence cannot be directly captured by word embeddings or language model, which are crucial for dependency parsing given its goal is to establish dependency relations between words. In this paper, we propose to implicitly model word connections by a word ordering model. The purpose of word ordering model is to generate a well-formed sentence given a bag of words. We human could make sentences easily from unordered words since we have syntactic knowledge, thus a model generating well-formed sentences from the bag of words encodes syntactic information. In addition, word ordering task allows us to use self-attention mechanism

[1]The tri-training approach accuracy on the tune set is 97.26% UAS (Weiss et al., 2015).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2857–2863
Brussels, Belgium, October 31 - November 4, 2018.

to model connections between words in the sentence. Different from the tri-training approach, our approach takes advantage of implicit word connections learned by self-attended word ordering model in an unsupervised way.

Experiments show that pre-trained word ordering model significantly improves our dependency parsing model. Ablation tests also show self-attention mechanism is critical. Moreover, by combining word ordering model and language model, our graph-based dependency parsing model achieves SOTA performance on the English Penn Treebank (Marcus et al., 1993) with 96.35% UAS and 95.25% LAS.

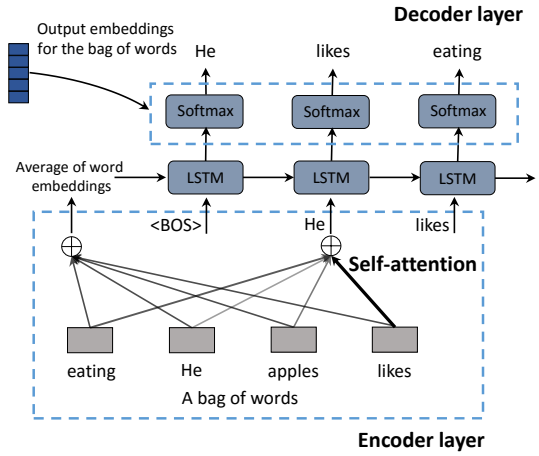

Figure 1: Overview of our word ordering model.

2 Neural Word Ordering Model

The target of word ordering is to generate a well-formed sentence given a bag of words. To capture word connections implicated in the sentence, an LSTM-based word ordering model with self-attention is proposed. Self-attention mechanism effectively decides which words in the word bag are more important in generating the next word. It improves the ability of our model to capture word connections. As illustrated in Figure 1, the proposed word ordering model consists of two layers:

Encoder Layer

Given a bag of words $w_1, w_2, ..., w_n$, we encode each word by a character-level BiLSTM ($c^{wo}_{w_{1:n}}$), which could reduce the parameters used in our model compared with word embeddings. For the input word of current time-step (w_i), a self-attention layer is utilized to align the word with its related words, producing its self-attended vector ($sa^{wo}_{w_i}$) as following:

$$s^i_j = \mathrm{v}^{\mathrm{T}}\mathrm{ReLU}(W^{sa}[c^{wo}_{w_i}; c^{wo}_{w_j}]) \quad (1)$$

$$a^i_t = \exp(s^i_t)/\Sigma^n_{j=1}\exp(s^i_j) \quad (2)$$

$$sa^{wo}_{w_i} = \Sigma^n_{j=1} a^i_j c^{wo}_{w_j} \quad (3)$$

The scores $s^i_{1:n}$ in self-attention explicitly represent the connections between words.

We then concatenate the character-level word embedding ($c^{wo}_{w_i}$) and its self-attended vector ($sa^{wo}_{w_i}$):

$$x^{wo}_i = [c^{wo}_{w_i}; sa^{wo}_{w_i}] \quad (4)$$

x^{wo}_i is fed into the decoder layer to generate the next word.

Decoder Layer

Given the current input vector (x^{wo}_i), which contains current word information and weighted information of related words, a forward LSTM is used to generate the next word. We initialize the forward LSTM with an average of the input word embeddings ($c^{wo}_{w_{i:n}}$). A virtual token $\langle$BOS$\rangle$ is added as the input of the first LSTM time-step:

$$\overrightarrow{h}^{wo}_i = \mathrm{LSTM}(x^{wo}_i, \overrightarrow{h}^{wo}_{i-1}) \quad (5)$$

At each time-step, the hidden state $\overrightarrow{h}^{wo}_i$ is utilized to predict the next word. Due to the output vocabulary is limited in the bag of words, we just compute scores for the given words ($w_{1:n}$):

$$so^i_j = \mathrm{v}^{\mathrm{T}}\mathrm{ReLU}(W^o[\overrightarrow{h}^{wo}_i; cd^{wo}_{w_j}]) + \overrightarrow{h}^{wo\mathrm{T}}_i M wd^{wo}_{w_j} \quad (6)$$

To reduce the parameters, each output word is represented by a character-level BiLSTM embedding ($cd^{wo}_{w_j}$) and a low-dimensional word embedding[2] ($wd^{wo}_{w_j}$). M is a matrix projecting a low-dimensional embedding back up to the dimensionality of LSTM hidden states. The scores $so^i_{1:n}$ are then normalized with Softmax, and the word with max probability is chosen as the next token.

The word ordering model could be trained easily in an unsupervised manner. Given a large set of unlabeled sentences, we can just ignore the word

[2]Character-level representations lack the capacity to differentiate between words that have very different meanings but that are spelled similarly. Low-dimensional word embeddings are added to improve the ability.

order of sentence and train the model to generate the corresponding well-formed sentence in the training set. To be specific, we minimize the sum of negative log probabilities of the ground truth words on the unlabeled data set. Different from language model, the choice for each decoder step is limited in the bag of words. Moreover, self-attention can be introduced into the word ordering model since we have known the bag of words, which could capture the dependency connections between words. We also pre-train a backward word ordering model to generate sentences in reverse order. The forward and backward models share character-level BiLSTM embeddings, self-attention layer, and Softmax layer.

Different from previous word ordering models (Liu et al., 2015; Schmaltz et al., 2016), self-attention mechanism is introduced into our model to capture word connections. Moreover, our more important goal is to implicitly utilize large-scale unlabeled data to help dependency parsing.

3 Neural Graph-based Parsing Model

We implement an LSTM-based neural network model as our graph-based dependency parsing baseline, which is similar to (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016). As shown in the Figure 2, it consists of three layers:

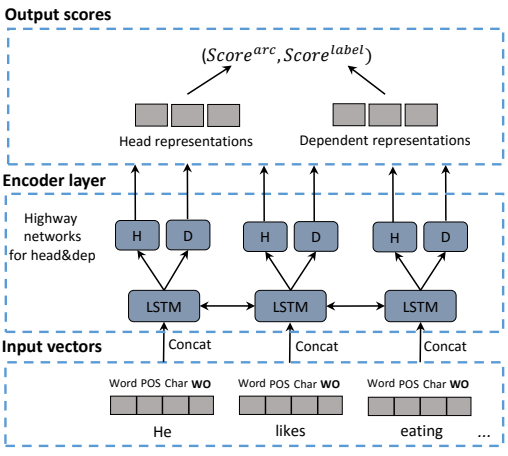

Figure 2: Overview of neural graph-based dependency parsing model. WO represents pre-trained vectors from word ordering model.

Input Layer

Given a n-words input sentence s with words $w_1, w_2, ..., w_n$ and its POS tags $p_1, p_2, ..., p_n$. The input layer creates a sequence of input vectors $x_{1:n}$ in which each x_i is a concatenation of its word embedding (e_{w_i}), POS tag embedding (e_{p_i}), character-level BiLSTM embedding (c_{w_i}), and word ordering model pre-trained vector (wo_{w_i}):

$$x_i = [e_{w_i}; e_{p_i}; c_{w_i}; wo_{w_i}] \quad (7)$$

To get the word ordering model pre-trained vector (wo_{w_i}), the sentence s is fed into the pre-trained word ordering model. Following Peters et al. (2018), we then combine the input vector ($x_i^{wo}=[c_{w_i}^{wo}; sa_{w_i}^{wo}]$) and L-layer BiLSTM vectors ($h_{i,j}^{wo}=[\overrightarrow{h}_{i,j}^{wo}; \overleftarrow{h}_{i,j}^{wo}] \mid j=1,2,...,L$) by a Softmax-normalized weight (W^{woc}) and a scalar parameter (γ):

$$wo_{w_i} = \gamma(W_0^{woc} x_i^{wo} + \Sigma_{j=1}^{L} W_j^{woc} h_{i,j}^{wo}) \quad (8)$$

The parameters of word ordering model are fixed during the training of parsing model. However, the weight and scalar parameters are tuned to better adapt to it. The combined output wo_{w_i} contains word connections from self-attention, word information from character-level embedding and contextual information from LSTM.

Encoder & Output Layer

To introduce more contextual information, we encode each input element by deep BiLSTMs:

$$v_i = \textbf{BiLSTM}(x_{1:n}, i) \quad (9)$$

Two different highway networks (Srivastava et al., 2015) are then used to encode head word representations ($v_{1:n}^{head}$) and dependent word representations ($v_{1:n}^{dep}$). For a head-dependent dependency pair (w_h, w_d), the dependency arc and label score are computed by two MLP networks:

$$i_{h,d} = [v_h^{head}; v_d^{dep}; v_h^{head} \odot v_d^{dep}] \quad (10)$$

$$s_{h,d}^{arc} = W_1^{arc} \mathrm{ReLU}(W_2^{arc} i_{h,d}) \quad (11)$$

$$s_{h,d}^{label} = W_1^{label} \mathrm{ReLU}(W_2^{label} i_{h,d}) \quad (12)$$

We use the Max-Margin criterion to train our parsing model, which is the same as (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016).

4 Experiments

4.1 Datasets

We conduct experiments on the English Penn Treebank and the CoNLL 09 English dataset. For

Method	PTB UAS / LAS	CONLL 09 UAS / LAS
Bohnet2012	- / -	92.87 / 90.60
Weiss2015	94.26 / 92.41	- / -
Alberti2015	94.23 / 92.36	92.7 / 90.56
Kiperwasser2016	93.9 / 91.9	- / -
Wang2016	94.08 / 91.82	- / -
Andor2016	94.61 / 92.79	93.22 / 91.23
Dozat2016	95.74 / 94.08	**95.21 / 93.20**
Smith2017	95.8 / 94.6	- / -
Choe2016	95.9 / 94.1	- / -
Baseline	95.12 / 93.98	93.61 / 91.70
Baseline+WO	95.66 / 94.54	94.21 / 92.27
Baseline+LM	96.05 / 94.88	94.50 / 92.70
Baseline+LM&WO	**96.35 / 95.25**	94.83 / 93.05

Table 1: Results on the English PTB dataset and CoNLL 09 English dataset. WO represents the pre-trained word ordering model. LM represents the pre-trained language model from Peters et al. (2018).

PTB dataset, we follow the standard splits. Using section 2-21 for training, section 22 as development set and 23 as test set. The treebank is converted to Stanford Basic Dependencies (Marneffe et al., 2006) by version 3.3.0[3] of the Stanford parser. The Stanford POS Tagger (Toutanova et al., 2003) is used for assigning POS tags. Following previous work, UAS (unlabeled attachment scores) and LAS (labeled attachment scores) are calculated by excluding punctuation. For the CoNLL 09 English dataset, we follow the standard practice and include all punctuation in the evaluation. We pre-train our word ordering model on the 1 billion word benchmark (Chelba et al., 2014).

4.2 Implementation Details

The graph-based dependency parsing model and word ordering model are optimized with Adam with an initial learning rate of $2e^{-3}$. The β_1 and β_2 used in Adam are 0.9 and 0.999 respectively.

The following hyper-parameters are used in all graph-based dependency parsing models: word embedding size = 300, POS tag embedding size = 32, character embedding size = 50, word-level LSTM hidden vector size = 200, word-level BiLSTM layer number = 3, character-level LSTM hidden vector size = 50, character-level BiLSTM layer number = 2, batch size = 32. We also apply dropout for the input and each layer with dropout rate of 0.3. We use pre-trained case-sensitive GloVe embeddings[4] to initialize word embeddings. These word embeddings are fine tuned with the graph-based dependency parsing model. The parameters of pre-trained word ordering model are fixed during the training of dependency parsing model. For deep BiLSTM, we concatenate the outputs of each layer as its final outputs.

For our word ordering model: input character-level LSTM hidden vector size = 512, input character-level BiLSTM layer number = 1, word-level LSTM hidden vector size = 1024, word-level LSTM layer number = 2, output character-level LSTM hidden vector size = 512, output character-level BiLSTM layer number = 1, output low-dimensional word embedding size = 64, batch size = 32, dropout for the input and each layer = 0.5.

4.3 Main Results & Ablation Study

Table 1 shows the performance of our model and previous work on two English benchmarks. Our model achieves promising results on both datasets. Two sets of experiments are provided to show the effectiveness of pre-trained word ordering model. Although our baseline system is similar to (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016) but with subtle differences in architecture, the baseline could perform much better to our surprise and thus constitutes a very strong baseline. Compared with this baseline, introducing the pre-trained word ordering model achieves a significant improvement (almost 0.6% UAS gains for both datasets, $p < 0.001$). To further show the effectiveness of word ordering model, we also implement an even stronger baseline with pre-trained language model[5]. Compared with this much stronger baseline, incorporating pre-trained word ordering model still achieves a significant improvement (0.3% UAS gains for both datasets, $p < 0.01$). We attribute the improvement to the ability of word ordering model to capture word connections, which cannot be directly captured by language model. Moreover, by combining with a pre-trained language model, our model outperforms current SOTA model from 95.9% UAS to 96.35% UAS on the PTB dataset. The introduction

[3] http://nlp.stanford.edu/software/lex-parser.shtml

[4] Downloaded from http://nlp.stanford.edu/data/glove.840B.300d.zip.

[5] We use the pre-trained language model provided by Peters et al. (2018), which can be downloaded at http://allennlp.org/elmo. The pre-trained language model vectors are added in the input layer, which are included in the same way as word ordering model. Peters et al. (2018) found the pre-trained language model works extremely well in six NLP tasks including QA, SRL, and others, we confirm its effectiveness in parsing task.

Model	UAS / LAS
Baseline+WO	**95.66 / 94.54**
-Self-attention vectors	95.38 / 94.27
Baseline+LM&WO	**96.35 / 95.25**
-Self-attention vectors	96.13 / 94.98

Table 2: Ablation tests of self-attention mechanism on the PTB dataset.

of POS tag features could contribute about 0.2% improvement in our experiments. The word ordering model could be more helpful without POS tag features and seem to compensate for the lack of POS tag features.

To show the importance of self-attention mechanism, we do ablation tests on the models with pre-trained word ordering model vectors. We remove self-attention vectors by replacing it with the character-level representations. As shown in table 2, self-attention further improves dependency parsing. Word connections modeled by self-attention are important for dependency parsing.

Figure 3 shows an example of word connections learned by the model, where we use the solid line to indicate the word connections learned by the word-ordering model and dashed line to the expected dependencies. We can see meaningful overlap could be observed in the example. The percentage of overlap between connections and dependency arcs is over 40% for the sentences less than 10 words. The differences between connections and dependency arcs are because that our word ordering model trained without any supervised dependency information. The connections are actually built to increase the likelihood.

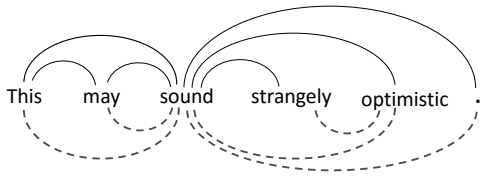

Figure 3: An example of self-attention. The solid line denotes the mostly attended word when generating the next word, dashed line denotes the correct dependency.

5 Conclusion

In this paper, we propose to implicitly capture word connections from large-scale unlabeled data by a word ordering model with self-attention. Experiments show these features are helpful for dependency parsing. Moreover, with the help of word ordering model and language model, our model achieves SOTA results on the PTB dataset. As for future work, we are testing on languages other than English.

Acknowledgement

We thank all the anonymous reviewers for their helpful comments. This work is supported by National Natural Science Foundation of China under Grant No.61876004 and No.61751201. The corresponding author of this paper is Baobao Chang.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1354–1359.

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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1455–1465.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *EMNLP-CoNLL*, pages 957–961.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural*

Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, pages 2331–2336.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

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*, pages 334–343.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 199–209.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 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. Association for Computational Linguistics.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 113–122.

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.

Marie Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. *Lrec*, pages 449–454.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*. Citeseer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *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*, pages 313–322.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1756–1765.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Allen Schmaltz, Alexander M. Rush, and Stuart M. Shieber. 2016. Word ordering without syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2319–2324.

Noah A. Smith, Chris Dyer, Miguel Ballesteros, Graham Neubig, Lingpeng Kong, and Adhiguna Kuncoro. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1249–1258.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

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.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2773–2781.

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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

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*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193.