Gelezen

An improved neural network model for joint POS tagging and dependency parsing

Dat Quoc Nguyen and **Karin Verspoor**
School of Computing and Information Systems
The University of Melbourne, Australia
{dqnguyen, karin.verspoor}@unimelb.edu.au

Abstract

We propose a novel neural network model for joint part-of-speech (POS) tagging and dependency parsing. Our model extends the well-known BIST graph-based dependency parser (Kiperwasser and Goldberg, 2016) by incorporating a BiLSTM-based tagging component to produce automatically predicted POS tags for the parser. On the benchmark English Penn treebank, our model obtains strong UAS and LAS scores at 94.51% and 92.87%, respectively, producing 1.5+% absolute improvements to the BIST graph-based parser, and also obtaining a state-of-the-art POS tagging accuracy at 97.97%. Furthermore, experimental results on parsing 61 "big" Universal Dependencies treebanks from raw texts show that our model outperforms the baseline UDPipe (Straka and Straková, 2017) with 0.8% higher average POS tagging score and 3.6% higher average LAS score. In addition, with our model, we also obtain state-of-the-art downstream task scores for biomedical event extraction and opinion analysis applications.

Our code is available together with all pretrained models at: https://github.com/datquocnguyen/jPTDP.

1 Introduction

Dependency parsing – a key research topic in natural language processing (NLP) in the last decade (Buchholz and Marsi, 2006; Nivre et al., 2007a; Kübler et al., 2009) – has also been demonstrated to be extremely useful in many applications such as relation extraction (Culotta and Sorensen, 2004; Bunescu and Mooney, 2005), semantic parsing (Reddy et al., 2016) and machine translation (Galley and Manning, 2009). In general, dependency parsing models can be categorized as graph-based (McDonald et al., 2005) and transition-based (Yamada and Matsumoto, 2003; Nivre, 2003). Most traditional graph- or transition-based models define a set of core and combined features (McDonald and Pereira, 2006; Nivre et al., 2007b; Bohnet, 2010; Zhang and Nivre, 2011), while recent state-of-the-art models propose neural network architectures to handle feature-engineering (Dyer et al., 2015; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017; Ma and Hovy, 2017).

Most traditional and neural network-based parsing models use automatically predicted POS tags as essential features. However, POS taggers are not perfect, resulting in error propagation problems. Some work has attempted to avoid using POS tags for dependency parsing (Dyer et al., 2015; Ballesteros et al., 2015; de Lhoneux et al., 2017), however, to achieve the strongest parsing scores these methods still require automatically assigned POS tags. Alternatively, joint POS tagging and dependency parsing has also attracted a lot of attention in NLP community as it could help improve both tagging and parsing results over independent modeling (Li et al., 2011; Hatori et al., 2011; Lee et al., 2011; Bohnet and Nivre, 2012; Zhang et al., 2015; Zhang and Weiss, 2016; Yang et al., 2018).

In this paper, we present a novel neural network-based model for jointly learning POS tagging and dependency paring. Our joint model extends the well-known BIST graph-based dependency parser (Kiperwasser and Goldberg, 2016) with an additional lower-level BiLSTM-based tagging component. In particular, this tagging component generates predicted POS tags for the parser component. Evaluated on the benchmark English Penn treebank test Section 23, our model pro-

Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 81–91
Brussels, Belgium, October 31 – November 1, 2018. ©2018 Association for Computational Linguistics
https://doi.org/10.18653/v1/K18-2008

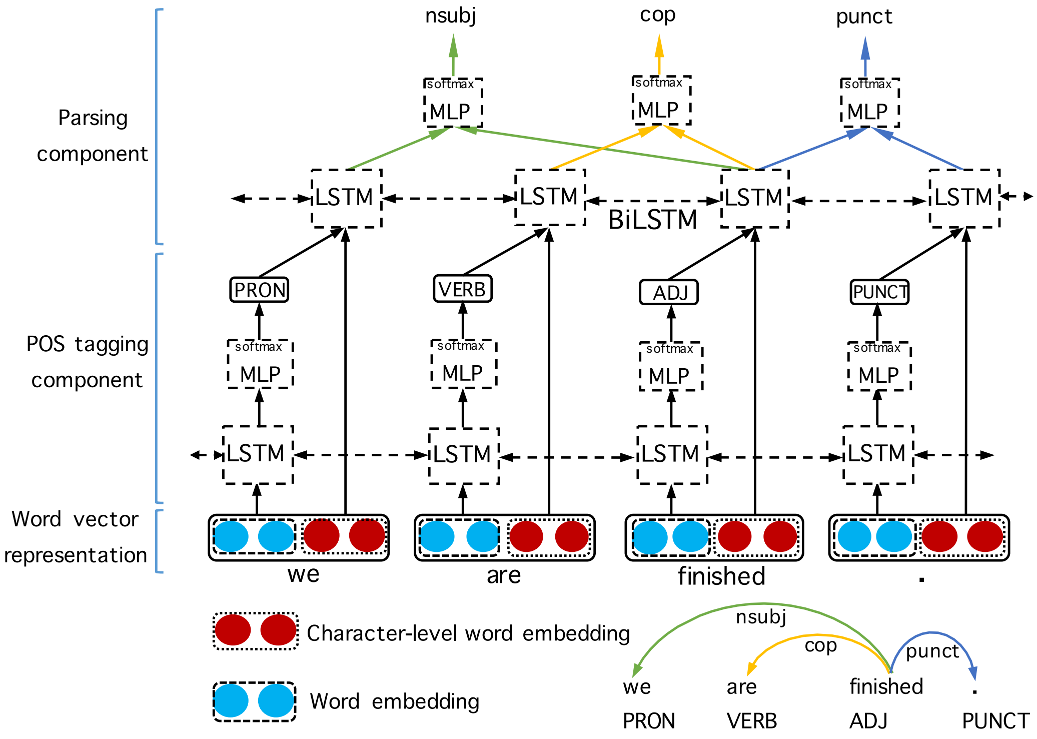

Figure 1: Illustration of our new model for joint POS tagging and graph-based dependency parsing.

duces a 1.5+% absolute improvement over the BIST graph-based parser with a strong UAS score of 94.51% and LAS score of 92.87%; and also obtaining a state-of-the-art POS tagging accuracy of 97.97%. In addition, multilingual parsing experiments from raw texts on 61 "big" Universal Dependencies treebanks (Zeman et al., 2018) show that our model outperforms the baseline UDPipe (Straka and Straková, 2017) with 0.8% higher average POS tagging score, 3.1% higher UAS and 3.6% higher LAS. Furthermore, experimental results on downstream task applications (Fares et al., 2018) show that our joint model helps produce state-of-the-art scores for biomedical event extraction and opinion analysis.

2 Our joint model

This section presents our model for joint POS tagging and graph-based dependency parsing. Figure 1 illustrates the architecture of our joint model which can be viewed as a two-component mixture of a tagging component and a parsing component. Given word tokens in an input sentence, the tagging component uses a BiLSTM to learn "latent" feature vectors representing these word tokens. Then the tagging component feeds these feature vectors into a multilayer perceptron with one hidden layer (MLP) to predict POS tags. The parsing component then uses another BiLSTM to learn another set of latent feature representations, based on both the input word tokens and the predicted POS tags. These latent feature representations are fed into a MLP to decode dependency arcs and another MLP to label the predicted dependency arcs.

2.1 Word vector representation

Given an input sentence s consisting of n word tokens $w_1, w_2, ..., w_n$, we represent each i^{th} word w_i in s by a vector $\mathbf{e}_i$. We obtain $\mathbf{e}_i$ by concatenating word embedding $\mathbf{e}^{(\mathrm{W})}_{w_i}$ and character-level word embedding $\mathbf{e}^{(\mathrm{C})}_{w_i}$:

$$\mathbf{e}_i = \mathbf{e}^{(\mathrm{W})}_{w_i} \circ \mathbf{e}^{(\mathrm{C})}_{w_i} \quad (1)$$

Here, each word type w in the training data is represented by a real-valued word embedding $\mathbf{e}^{(\mathrm{W})}_{w}$. Given the word type w consisting of k characters $w = c_1c_2...c_k$ where each jth character in w is represented by a character embedding $\mathbf{c}_j$, we use a sequence BiLSTM (BiLSTM$_{\text{seq}}$) to learn its character-level vector representation (Ballesteros et al., 2015; Plank et al., 2016). The input to BiLSTM$_{\text{seq}}$ is the sequence of k character embeddings $\mathbf{c}_{1:k}$, and the output is a concatenation of outputs of a forward LSTM (LSTM$_{\text{f}}$) reading the input in its regular order and a reverse LSTM (LSTM_r) reading the input in reverse:

$$\mathbf{e}_w^{(\text{C})} = \text{BiLSTM}_\text{seq}(\mathbf{c}_{1:k}) \ = \text{LSTM}_\text{f}(\mathbf{c}_{1:k}) \circ \text{LSTM}_\text{r}(\mathbf{c}_{k:1})$$

2.2 Tagging component

We feed the sequence of vectors $\mathbf{e}_{1:n}$ with an additional context position index i into another BiLSTM (BiLSTM_pos), resulting in latent feature vectors $\boldsymbol{v}_i^{(\text{pos})}$ each representing the i^th word w_i in s:

$$\boldsymbol{v}_i^{(\text{pos})} = \text{BiLSTM}_\text{pos}(\mathbf{e}_{1:n}, i) \quad (2)$$

We use a MLP with softmax output (MLP_pos) on top of the BiLSTM_pos to predict POS tag of each word in s. The number of nodes in the output layer of this MLP_pos is the number of POS tags. Given $\boldsymbol{v}_i^{(\text{pos})}$, we compute an output vector as:

$$\boldsymbol{\vartheta}_i = \text{MLP}_\text{pos}(\boldsymbol{v}_i^{(\text{pos})}) \quad (3)$$

Based on output vectors $\boldsymbol{\vartheta}_i$, we then compute the cross-entropy objective loss $\mathcal{L}_\textbf{POS}(\hat{\mathbf{t}}, \mathbf{t})$, in which $\hat{\mathbf{t}}$ and $\mathbf{t}$ are the sequence of predicted POS tags and sequence of gold POS tags of words in the input sentence s, respectively (Goldberg, 2016). Our tagging component thus can be viewed as a simplified version of the POS tagging model proposed by Plank et al. (2016), without their additional auxiliary loss for rare words.

2.3 Parsing component

Assume that $p_1, p_2, ..., p_n$ are the predicted POS tags produced by the tagging component for the input words. We represent each i^{th} predicted POS tag by a vector embedding $\mathbf{e}_{p_i}^{(\text{P})}$. We then create a sequence of vectors $\boldsymbol{x}_{1:n}$ in which each $\boldsymbol{x}_i$ is produced by concatenating the POS tag embedding $\mathbf{e}_{p_i}^{(\text{P})}$ and the word vector representation $\mathbf{e}_i$:

$$\mathbf{x}_i = \mathbf{e}_{p_i}^{(\text{P})} \circ \mathbf{e}_i = \mathbf{e}_{p_i}^{(\text{P})} \circ \mathbf{e}_{w_i}^{(\text{W})} \circ \mathbf{e}_{w_i}^{(\text{C})} \quad (4)$$

We feed the sequence of vectors $\mathbf{x}_{1:n}$ with an additional index i into a BiLSTM (BiLSTM_dep), resulting in latent feature vectors $\boldsymbol{v}_i$ as follows:

$$\boldsymbol{v}_i = \text{BiLSTM}_\text{dep}(\mathbf{x}_{1:n}, i) \quad (5)$$

Based on latent feature vectors $\boldsymbol{v}_i$, we follow a common arc-factored parsing approach to decode dependency arcs (McDonald et al., 2005). In particular, a dependency tree can be formalized as a directed graph. An arc-factored parsing approach learns the scores of the arcs in the graph (Kübler et al., 2009). Here, we score an arc by using a MLP with a one-node output layer (MLP_arc) on top of the BiLSTM_dep:

$$\begin{aligned} &\text{score}_\text{arc}(i, j) && (6) \\ &\quad = \text{MLP}_\text{arc}\big(\boldsymbol{v}_i \circ \boldsymbol{v}_j \circ (\boldsymbol{v}_i * \boldsymbol{v}_j) \circ |\boldsymbol{v}_i - \boldsymbol{v}_j|\big) \end{aligned}$$

where $(\boldsymbol{v}_i * \boldsymbol{v}_j)$ and $|\boldsymbol{v}_i - \boldsymbol{v}_j|$ denote the element-wise product and the absolute element-wise difference, respectively; and $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are correspondingly the latent feature vectors associating to the i^{th} and j^{th} words in s, computed by Equation 5.

Given the arc scores, we use the Eisner (1996)'s decoding algorithm to find the highest scoring projective parse tree:

$$\text{score}(s) = \underset{\hat{y} \in \mathcal{Y}(s)}{\text{argmax}} \sum_{(h,m) \in \hat{y}} \text{score}_\text{arc}(h, m) \quad (7)$$

where $\mathcal{Y}(s)$ is the set of all possible dependency trees for the input sentence s while $\text{score}_\text{arc}(h, m)$ measures the score of the arc between the head h^th word and the modifier m^th word in s.

Following Kiperwasser and Goldberg (2016), we compute a margin-based hinge loss $\mathcal{L}_\textbf{ARC}$ with loss-augmented inference to maximize the margin between the gold unlabeled parse tree and the highest scoring incorrect tree.

For predicting dependency relation type of a head-modifier arc, we use another MLP with softmax output (MLP_rel) on top of the BiLSTM_dep. Here, the number of the nodes in the output layer of this MLP_rel is the number of dependency relation types. Given an arc (h, m), we compute an output vector as:

$$\begin{aligned} &\mathbf{v}_{(h,m)} && (8) \\ &\quad = \text{MLP}_\text{rel}\big(\boldsymbol{v}_h \circ \boldsymbol{v}_m \circ (\boldsymbol{v}_h * \boldsymbol{v}_m) \circ |\boldsymbol{v}_h - \boldsymbol{v}_m|\big) \end{aligned}$$

Based on output vectors $\mathbf{v}_{(h,m)}$, we also compute another cross-entropy objective loss $\mathcal{L}_\textbf{REL}$ for relation type prediction, using only the gold labeled parse tree.

Our parsing component can be viewed as an extension of the BIST graph-based dependency model (Kiperwasser and Goldberg, 2016), where we additionally incorporate the character-level vector representations of words.

2.4 Joint model training

The training objective loss of our joint model is the sum of the POS tagging loss $\mathcal{L}_\text{POS}$, the structure loss $\mathcal{L}_\text{ARC}$ and the relation labeling loss $\mathcal{L}_\text{REL}$:

$$\mathcal{L} = \mathcal{L}_\text{POS} + \mathcal{L}_\text{ARC} + \mathcal{L}_\text{REL} \quad (9)$$

The model parameters, including word embeddings, character embeddings, POS embeddings, three one-hidden-layer MLPs and three BiLSTMs, are learned to minimize the sum $\mathcal{L}$ of the losses.

Most neural network-based joint models for POS tagging and dependency parsing are transition-based approaches (Alberti et al., 2015; Zhang and Weiss, 2016; Yang et al., 2018), while our model is a graph-based method. In addition, the joint model JMT (Hashimoto et al., 2017) defines its dependency parsing task as a head selection task which produces a probability distribution over possible heads for each word (Zhang et al., 2017).

Our model is the successor of the joint model jPTDP v1.0 (Nguyen et al., 2017) which is also a graph-based method. However, unlike our model, jPTDP v1.0 uses a BiLSTM to learn "shared" latent feature vectors which are then used for both POS tagging and dependency parsing tasks, rather than using two separate layers. As mentioned in Section 4, our model generally outperforms jPTDP v1.0 with 2.5+% LAS improvements on universal dependencies (UD) treebanks.

2.5 Implementation details

Our model is released as jPTDP v2.0, available at https://github.com/datquocnguyen/jPTDP. Our jPTDP v2.0 is implemented using DyNet v2.0 (Neubig et al., 2017) with a fixed random seed.[1] Word embeddings are initialized either randomly or by pre-trained word vectors, while character and POS tag embeddings are randomly initialized. For learning character-level word embeddings, we use one-layer $\text{BiLSTM}_{\text{seq}}$, and set the size of LSTM hidden states to be equal to the vector size of character embeddings.

We apply dropout (Srivastava et al., 2014) with a 67% keep probability to the inputs of BiLSTMs and MLPs. Following Iyyer et al. (2015) and Kiperwasser and Goldberg (2016), we also apply *word dropout* to learn an embedding for unknown words: we replace each word token w appearing $\#(w)$ times in the training set with a special "unk" symbol with probability $\mathsf{p}_{unk}(w) = \frac{0.25}{0.25+\#(w)}$. This procedure only involves the word embedding part in the input word vector representation.

We optimize the objective loss using Adam (Kingma and Ba, 2014) with an initial learning rate at 0.001 and no mini-batches. For training, we run for 30 epochs, and restart the Adam optimizer and anneal its initial learning rate at a proportion of 0.5 every 10 epochs. We evaluate the *mixed accuracy* of correctly assigning POS tag together with dependency arc and relation type on the development set after each training epoch. We choose the model with the highest mixed accuracy on the development set, which is then applied to the test set for the evaluation phase.

For all experiments presented in this paper, we use 100-dimensional word embeddings, 50-dimensional character embeddings and 100-dimensional POS tag embeddings. We also fix the number of hidden nodes in MLPs at 100. Due to limited computational resource, for experiments presented in Section 3, we perform a minimal grid search of hyper-parameters to select the number of $\text{BiLSTM}_{\text{pos}}$ and $\text{BiLSTM}_{\text{dep}}$ layers from $\{1, 2\}$ and the size of LSTM hidden states in each layer from $\{128, 256\}$. For experiments presented in sections 4 and 5, we fix the number of BiLSTM layers at 2 and the size of hidden states at 128.

3 Experiments on English Penn treebank

Experimental setup: We evaluate our model using the English WSJ Penn treebank (Marcus et al., 1993). We follow a standard data split to use sections 02-21 for training, Section 22 for development and Section 23 for test (Chen and Manning, 2014), employing the Stanford conversion toolkit v3.3.0 to generate dependency trees with Stanford basic dependencies (de Marneffe and Manning, 2008).

Word embeddings are initialized by 100-dimensional GloVe word vectors pre-trained on Wikipedia and Gigaword (Pennington et al., 2014).[2] As mentioned in Section 2.5, we perform a minimal grid search of hyper-parameters and find that the highest mixed accuracy on the development set is obtained when using 2 BiLSTM layers and 256-dimensional LSTM hidden states (in Table 1, we present scores obtained on the development set when using 2 BiLSTM layers).

Main results: Table 2 compares our UAS and LAS scores on the test set with previous published results in terms of the dependency annotations.[3]

[1] https://github.com/clab/dynet

[2] https://nlp.stanford.edu/projects/glove

[3] Choe and Charniak (2016) reported the highest UAS score at 95.9% and LAS score at 94.1% to date on the test set, using the Stanford conversion toolkit v3.3.0 to convert the output constituent trees into dependency representations.

#states	With punctuations			Without pun.	
	POS	UAS	LAS	UAS	LAS
128	**97.64**	93.68	92.11	94.42	92.61
256	97.63	**93.89**	**92.33**	**94.63**	**92.82**
Chen and Manning (2014)				92.0	89.7
Dyer et al. (2015)				93.2	90.9
BIST-graph [K&G16]				93.3	91.0
Zhang et al. (2017)				94.30	91.95
Ma and Hovy (2017)				94.77	92.66
Dozat and Manning (2017)				**95.24**	**93.37**

Table 1: Results on the development set. **#states** and "**Without pun.**" denote the size of LSTM hidden states and the scores computed without punctuations, respectively. "POS" indicates the POS tagging accuracy. [K&G16] denotes results reported in Kiperwasser and Goldberg (2016).

Model	POS	UAS	LAS
Chen and Manning (2014)	97.3	91.8	89.6
Dyer et al. (2015)	97.3	93.1	90.9
Weiss et al. (2015)	97.44	93.99	92.05
BIST-graph [K&G16]	97.3	93.1	91.0
BIST-transition [K&G16]	97.3	93.9	91.9
Kuncoro et al. (2016)	97.3	94.26	92.06
Andor et al. (2016)	97.44	94.61	92.79
Zhang et al. (2017)	97.3	94.10	91.90
Ma and Hovy (2017)	97.3	94.88	92.98
Dozat and Manning (2017)	97.3	**95.44**	**93.76**
Dozat and Manning (2017) [•]	97.3	95.66	94.03
Bohnet and Nivre (2012) [⋆]	97.42	93.67	92.68
Alberti et al. (2015)	97.44	94.23	92.36
Zhang and Weiss (2016)	–	93.43	91.41
Hashimoto et al. (2017)	–	**94.67**	**92.90**
Yang et al. (2018)	97.54	94.18	92.26
Our model	**97.97**	94.51	92.87

Table 2: Results on the test set. POS tagging accuracies are computed on all tokens. UAS and LAS are computed without punctuations. [•]: the treebank was converted with the Stanford conversion toolkit v3.5.0. [⋆]: the treebank was converted with the head rules of Yamada and Matsumoto (2003). For both [•] and [⋆], obtained parsing scores are just for reference, not for comparison.

The first 11 rows present scores of dependency parsers in which POS tags were predicted by using an external POS tagger such as the Stanford tagger (Toutanova et al., 2003). The last 6 rows present scores for joint models. Clearly, our model produces very competitive parsing results. In particular, our model obtains a UAS score at 94.51% and a LAS score at 92.87% which are about 1.4% and 1.9% absolute higher than UAS and LAS scores of the BIST graph-based model (Kiperwasser and Goldberg, 2016), respectively. Our model also does better than the previous transition-based joint models in Alberti et al. (2015), Zhang and Weiss (2016) and Yang et al. (2018), while obtaining similar UAS and LAS scores to the joint model JMT proposed by Hashimoto et al. (2017).

We achieve 0.9% lower parsing scores than the state-of-the-art dependency parser of Dozat and Manning (2017). While also a BiLSTM- and graph-based model, it uses a more sophisticated attention mechanism "*biaffine*" for better decoding dependency arcs and relation types. In future work, we will extend our model with the *biaffine* attention mechanism to investigate the benefit for our model. Other differences are that they use a higher dimensional representation than ours, but rely on predicted POS tags.

We also obtain a state-of-the-art POS tagging accuracy at 97.97% on the test Section 23, which is about 0.4+% higher than those by Bohnet and Nivre (2012), Alberti et al. (2015) and Yang et al. (2018). Other previous joint models did not mention their specific POS tagging accuracies.[4]

4 UniMelb in the CoNLL 2018 shared task on UD parsing

Our UniMelb team participated with jPTDP v2.0 in the CoNLL 2018 shared task on parsing 82 treebank test sets (in 57 languages) from raw text to universal dependencies (Zeman et al., 2018). The 82 treebanks are taken from UD v2.2 (Nivre et al., 2018), where 61/82 test sets are for "big" UD treebanks for which both training and development data sets are available and 5/82 test sets are extra "parallel" test sets in languages where another big treebank exists. In addition, 7/82 test sets are for "small" UD treebanks for which development data is not available. The remaining 9/82 sets are in low-resource languages without training data or with a few gold-annotation sample sentences.

For the 7 small treebanks without development data available, we split training data into two parts with a ratio 9:1, and then use the larger part for training and the smaller part for development. For each big or small treebank, we train a joint model for *universal* POS tagging and dependency parsing, using a fixed random seed and a fixed set

[4] Hashimoto et al. (2017) showed that JMT obtains a POS tagging accuracy of 97.55% on WSJ sections 22-24.

	System	All (82)	Big (61)	PUD (5)	Small (7)	Low (9)
UPOS	UDPipe 1.2	87.32	93.71	85.23	**87.36**	45.20
	UniMelb	**87.90**	**94.50**	**85.33**	87.12	45.20
	goldseg.	_	95.63	90.21	87.64	_
UAS	UDPipe 1.2	71.64	78.78	71.22	63.17	30.08
	UniMelb	**74.16**	**81.83**	**73.17**	**64.71**	30.08
	goldseg.	_	85.01	81.81	67.46	_
LAS	UDPipe 1.2	65.80	74.14	66.63	55.01	17.17
	UniMelb	**68.65**	**77.69**	**68.72**	**56.12**	17.17
	goldseg.	_	80.68	75.03	58.65	_

Table 3: Official macro-average F1 scores computed on all tokens for UniMelb and the baseline UDPipe 1.2 in the CoNLL 2018 shared task on UD parsing from raw texts (Zeman et al., 2018). "UPOS" denotes the universal POS tagging score. "All", "Big", "PUD", "Small" and "Low" refer to the macro-average scores over all 81, 61 big treebank, 5 parallel, 7 small treebank and 9 low-resource treebank test sets, respectively. "goldseg." denotes the scores of our jPTDP v2.0 model regarding gold segmentation, detailed in Table 4.

of hyper-parameters as mentioned in Section 2.5.[5] We evaluate the mixed accuracy on the development set after each training epoch, and select the model with the highest mixed accuracy.

For parsing from raw text to universal dependencies, we employ CoNLL-U test files pre-processed by the baseline UDPipe 1.2 (Straka and Straková, 2017). Here, we utilize the tokenization, word and sentence segmentation predicted by UDPipe 1.2. For 68 big and small treebank test files, we use the corresponding trained joint models. We use the joint models trained for *cs_pdt*, *en_ewt*, *fi_tdt*, *ja_gsd* and *sv_talbanken* to process 5 parallel test files *cs_pud*, *en_pud*, *fi_pud*, *ja_modern* and *sv_pud*, respectively. Since we do not focus on low-resource languages, we employ the baseline UDPipe 1.2 to process 9 low-resource treebank test files. The final test runs are carried out on the TIRA platform (Potthast et al., 2014).

Table 3 presents our results in the CoNLL 2018 shared task on multilingual parsing from raw texts to universal dependencies (Zeman et al., 2018). Over all 82 test sets, we outperform the baseline UDPipe 1.2 with 0.6% absolute higher average UPOS F1 score and 2.5+% higher average UAS and LAS F1 scores. In particular, for the "big" category consisting of 61 treebank test sets, we obtain 0.8% higher UPOS and 3.1% higher UAS and 3.6% higher LAS than UDPipe 1.2.

Our (UniMelb) official LAS-based rank is at 14^{th} place while the baseline UDPipe 1.2 is at 18^{th} place over total 26 participating systems.[6] However, it is difficult to make a clear comparison between our jPTDP v2.0 and the parsing models used in other top systems. Several better participating systems simply reuse the state-of-the-art *biaffine* dependency parser (Dozat and Manning, 2017; Dozat et al., 2017), constructing ensemble models or developing treebank concatenation strategies to obtain larger training data, which is likely to produce better scores than ours (Zeman et al., 2018).

Recall that the shared task focuses on parsing from raw texts. Most higher-ranking systems aim to improve the pre-processing steps of tokenization[7], word[8] and sentence[9] segmentation, resulting in significant improvements in final parsing scores. For example, in the CoNLL 2017 shared task on UD parsing (Zeman et al., 2017), UDPipe 1.2 obtained 0.1+% higher average tokenization and word segmentation scores and 0.2% higher average sentence segmentation score than UDPipe 1.1, resulting in 1+% improvement in the final average LAS F1 score while both UDPipe 1.2 and UDPipe 1.1 shared exactly the same remaining components. Utilizing better pre-processors, as used in other participating systems, should likewise improve our final parsing scores.

In Table 3, we also present our average UPOS, UAS and LAS accuracies with respect to (w.r.t.) gold-standard tokenization, word and sentence segmentation. For more details and future comparison, Table 4 presents the UPOS, UAS and LAS scores w.r.t. gold-standard segmentation, obtained by jPTDP v2.0 on each UD v2.2–CoNLL 2018 shared task test set. Compared to the scores presented in Table 3 in Nguyen et al. (2017) on overlapped treebanks, our model jPTDP v2.0 generally produces 2.5+% improvements in UAS and LAS scores to jPTDP v1.0 (Nguyen et al., 2017).

[5] We initialize word embeddings by 100-dimensional pre-trained vectors from Ginter et al. (2017). For a language where pre-trained word vectors are not available in Ginter et al. (2017), word embeddings are randomly initialized.

[6] http://universaldependencies.org/conll18/results.html

[7] http://universaldependencies.org/conll18/results-tokens.html

[8] http://universaldependencies.org/conll18/results-words.html

[9] http://universaldependencies.org/conll18/results-sentences.html

Treebank	Code	UPOS	UAS	LAS	Treebank	Code	UPOS	UAS	LAS
Afrikaans-AfriBooms	af_afribooms	95.73	82.57	78.89	Italian-ISDT	it_isdt	98.01	92.33	90.20
Ancient_Greek-PROIEL	grc_proiel	96.05	77.57	72.84	Italian-PoSTWITA	it_postwita	95.41	84.20	79.11
Ancient_Greek-Perseus	grc_perseus	88.95	65.09	58.35	Japanese-GSD	ja_gsd	97.27	94.21	92.02
Arabic-PADT	ar_padt	96.33	86.08	80.97	Japanese-Modern **[p]**	ja_modern	70.53	66.88	49.51
Basque-BDT	eu_bdt	93.62	79.86	75.07	Korean-GSD	ko_gsd	93.35	81.32	76.58
Bulgarian-BTB	bg_btb	98.07	91.47	87.69	Korean-Kaist	ko_kaist	93.53	83.59	80.74
Catalan-AnCora	ca_ancora	98.46	90.78	88.40	Latin-ITTB	la_ittb	98.12	82.99	79.96
Chinese-GSD	zh_gsd	93.26	82.50	77.51	Latin-PROIEL	la_proiel	95.54	74.95	69.76
Croatian-SET	hr_set	97.42	88.74	83.62	Latin-Perseus **[s]**	la_perseus	82.36	57.21	46.28
Czech-CAC	cs_cac	98.87	89.85	87.13	Latvian-LVTB	lv_lvtb	93.53	81.06	76.13
Czech-FicTree	cs_fictree	97.98	88.94	85.64	North_Sami-Giella **[s]**	sme_giella	87.48	65.79	58.09
Czech-PDT	cs_pdt	98.74	89.64	87.04	Norwegian-Bokmaal	no_bokmaal	97.73	89.83	87.57
Czech-PUD **[p]**	cs_pud	96.71	87.62	82.28	Norwegian-Nynorsk	no_nynorsk	97.33	89.73	87.29
Danish-DDT	da_ddt	96.18	82.17	78.88	Norwegian-NynorskLIA **[s]**	no_nynorsklia	85.22	64.14	54.31
Dutch-Alpino	nl_alpino	95.62	86.34	82.37	Old_Church_Slavonic-PROIEL	cu_proiel	93.69	80.59	73.93
Dutch-LassySmall	nl_lassysmall	95.21	86.46	82.14	Old_French-SRCMF	fro_srcmf	95.12	86.65	81.15
English-EWT	en_ewt	95.48	87.55	84.71	Persian-Seraji	fa_seraji	96.66	88.07	84.07
English-GUM	en_gum	94.10	84.88	80.45	Polish-LFG	pl_lfg	98.22	95.29	93.10
English-LinES	en_lines	95.55	80.34	75.40	Polish-SZ	pl_sz	97.05	90.98	87.66
English-PUD **[p]**	en_pud	95.25	87.49	84.25	Portuguese-Bosque	pt_bosque	96.76	88.67	85.71
Estonian-EDT	et_edt	96.87	85.45	82.13	Romanian-RRT	ro_rrt	97.43	88.74	83.54
Finnish-FTB	fi_ftb	94.53	86.10	82.45	Russian-SynTagRus	ru_syntagrus	98.51	91.00	88.91
Finnish-PUD **[p]**	fi_pud	96.44	87.54	84.60	Russian-Taiga **[s]**	ru_taiga	85.49	65.52	56.33
Finnish-TDT	fi_tdt	96.12	86.07	82.92	Serbian-SET	sr_set	97.40	89.32	85.03
French-GSD	fr_gsd	97.11	89.45	86.43	Slovak-SNK	sk_snk	95.18	85.88	81.89
French-Sequoia	fr_sequoia	97.92	89.71	87.43	Slovenian-SSJ	sl_ssj	97.79	88.26	86.10
French-Spoken	fr_spoken	94.25	79.80	73.45	Slovenian-SST	sl_sst **[s]**	89.50	66.14	58.13
Galician-CTG	gl_ctg	97.12	85.09	81.93	Spanish-AnCora	es_ancora	98.57	90.30	87.98
Galician-TreeGal **[s]**	gl_treegal	93.66	77.71	71.63	Swedish-LinES	sv_lines	95.51	83.60	78.97
German-GSD	de_gsd	94.07	81.45	76.68	Swedish-PUD **[p]**	sv_pud	92.10	79.53	74.53
Gothic-PROIEL	got_proiel	93.45	79.80	71.85	Swedish-Talbanken	sv_talbanken	96.55	86.53	83.01
Greek-GDT	el_gdt	96.59	87.52	84.64	Turkish-IMST	tr_imst	92.93	70.53	62.55
Hebrew-HTB	he_htb	96.24	87.65	82.64	Ukrainian-IU	uk_iu	95.24	83.47	79.38
Hindi-HDTB	hi_hdtb	96.94	93.25	89.83	Urdu-UDTB	ur_udtb	93.35	86.74	80.44
Hungarian-Szeged	hu_szeged	92.07	76.18	69.75	Uyghur-UDT	ug_udt	87.63	76.14	63.37
Indonesian-GSD	id_gsd	93.29	84.64	77.71	Vietnamese-VTB	vi_vtb	87.63	67.72	58.27
Irish-IDT **[s]**	ga_idt	89.74	75.72	65.78		**Average**	94.49	83.11	78.18

Table 4: UPOS, UAS and LAS scores computed on all tokens of our jPTDP v2.0 model regarding gold-standard segmentation on 73 CoNLL-2018 shared task test sets "Big", "PUD" and "Small" – UD v2.2 (Nivre et al., 2018). **[p]** and **[s]** denote the "PUD" extra parallel and small test sets, respectively. For each treebank, a joint model is trained using a fixed set of hyper-parameters as mentioned in Section 2.5.

5 UniMelb in the EPE 2018 campaign

Our UniMelb team also participated with jPTDP v2.0 in the 2018 Extrinsic Parser Evaluation (EPE) campaign (Fares et al., 2018).[10] The EPE 2018 campaign runs in collaboration with the CoNLL 2018 shared task, which aims to evaluate dependency parsers by comparing their performance on three downstream tasks: biomedical event extraction (Björne et al., 2017), negation resolution (Lapponi et al., 2017) and opinion analysis (Johansson, 2017). Here, participants only need to provide parsing outputs of English raw texts used in these downstream tasks; the campaign organizers then compute end-to-end downstream task scores. General background can be also found in the first EPE edition 2017 (Oepen et al., 2017).

Unlike EPE 2017, the EPE 2018 campaign limited the training data to the English UD treebanks only. We unfortunately were unaware of this restriction during development of our model. Thus, we trained a jPTDP v2.0 model on dependency trees generated with the Stanford basic dependencies on a combination of the WSJ treebank, sections 02-21, and the training split of the GENIA treebank (Tateisi et al., 2005). We used the fixed set of hyper-parameters as used for the CoNLL 2018 shared task as mentioned in Section 2.5.[11] We then submitted the parsing outputs by run-

[10] http://epe.nlpl.eu

[11] Word embeddings are initialized by the 100-dimensional pre-trained GloVe word vectors.

Task	Development set				Evaluation set			
	Pre.	Rec.	F1	SP17	Pre.	Rec.	F1	SP17
Event extraction	57.87	51.20	**54.33**$_{1}$	52.67$_{54.59}$	58.52	49.43	**53.59**$_{1}$	50.29$_{50.23}$
Negation resolution	100.0	44.51	61.60$_{3}$	**64.85**$_{65.37}$	100.0	41.83	58.99$_{3}$	**65.13**$_{66.16}$
Opinion analysis	69.12	64.65	**66.81**$_{1}$	66.63$_{68.53}$	66.67	62.88	**64.72**$_{1}$	63.72$_{65.14}$
Average	–	–	60.91$_{1}$	**61.38**$_{62.83}$	–	–	59.10$_{1}$	**59.71**$_{60.51}$

Table 5: Downstream task scores Precision (Prec.), Recall (Rec.) and F1 for our UniMelb team. The *subscript* in the F1 column denotes the unofficial rank of UniMelb over 17 participating teams at EPE 2018 (Fares et al., 2018). "SP17" denotes the F1 scores obtained by the EPE 2017 system Stanford-Paris (Schuster et al., 2017) with respect to (w.r.t.) the Stanford basic dependencies. The *subscript* in the SP17 column denotes the F1 scores obtained by Stanford-Paris w.r.t. the UD-v1-enhanced type of dependency representations, in which the average F1 score at 60.51 is the highest one at EPE 2017.

ning our trained model on the pre-processed tokenized and sentence-segmented data provided by the campaign on the TIRA platform.

Table 5 presents the results we obtained for three downstream tasks at EPE 2018 (Fares et al., 2018). Since we employed external training data, our obtained scores are not officially ranked. In total 17 participating teams, we obtained the highest average F1 score over the three downstream tasks (i.e., we ranked first, unofficially). In particular, we achieved the highest F1 scores for both biomedical event extraction and opinion analysis. Our results may be high because the training data we used is larger than the English UD treebanks used by other teams.

Table 5 also presents scores from the Stanford-Paris team (Schuster et al., 2017)—the first-ranked team at EPE 2017 (Oepen et al., 2017). Both EPE 2017 and 2018 campaigns use the same downstream task setups, therefore the downstream task scores are directly comparable. Note that Stanford-Paris employed the state-of-the-art *biaffine* dependency parser (Dozat et al., 2017) with larger training data. In particular, Stanford-Paris not only used the WSJ sections 02-21 and the training split of the GENIA treebank (as we did), but also included the Brown corpus. The downstream application of negation resolution requires parsing of fiction, which is one the genres included in the Brown corpus. Hence it is reasonable that the Stanford-Paris team produced better negation resolution scores than we did.

However, in terms of the Stanford basic dependencies, while we employ a less accurate parsing model with smaller training data, we obtain higher downstream task scores for event extraction and opinion analysis than the Stanford-Paris team. Consequently, better intrinsic parsing performance does not always imply better extrinsic downstream application performance. Similar observations on the biomedical event extraction and opinion analysis tasks can also be found in Nguyen and Verspoor (2018) and Gómez-Rodríguez et al. (2017), respectively. Further investigations of this pattern requires much deeper understanding of the architecture of the downstream task systems, which is left for future work.

6 Conclusion

In this paper, we have presented a novel neural network model for joint POS tagging and graph-based dependency parsing. On the benchmark English WSJ Penn treebank, our model obtains strong parsing scores UAS at 94.51% and LAS at 92.87%, and a state-of-the-art POS tagging accuracy at 97.97%.

We also participated with our joint model in the CoNLL 2018 shared task on multilingual parsing from raw texts to universal dependencies, and obtained very competitive results. Specifically, using the same CoNLL-U files pre-processed by UDPipe (Straka and Straková, 2017), our model produced 0.8% higher POS tagging, 3.1% higher UAS and 3.6% higher LAS scores on average than UDPipe on 61 big UD treebank test sets. Furthermore, our model also helps obtain state-of-the-art downstream task scores for the biomedical event extraction and opinion analysis applications.

We believe our joint model can serve as a new strong baseline for both intrinsic POS tagging and dependency parsing tasks as well as for extrinsic downstream applications. Our code and pre-trained models are available at: `https://github.com/datquocnguyen/jPTDP`.

Acknowledgments

This work was supported by the ARC Discovery Project DP150101550 and ARC Linkage Project LP160101469.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved Transition-Based Parsing and Tagging with Neural Networks. In *Proceedings of EMNLP*. 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 ACL*. pages 2442–2452.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In *Proceedings of EMNLP*. pages 349–359.

Jari Björne, Filip Ginter, and Tapio Salakoski. 2017. EPE 2017: The Biomedical event extraction downstream application. In *Proceedings of the EPE 2017 Shared Task*. pages 17–24.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING*. pages 89–97.

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 EMNLP-CoNLL*. pages 1455–1465.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*. pages 149–164.

Razvan Bunescu and Raymond Mooney. 2005. A Shortest Path Dependency Kernel for Relation Extraction. In *Proceedings of HLT/EMNLP*. pages 724–731.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*. pages 740–750.

Do Kook Choe and Eugene Charniak. 2016. Parsing as Language Modeling. In *Proceedings of EMNLP*. pages 2331–2336.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of ACL*. pages 423–429.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017. From Raw Text to Universal Dependencies - Look, No Tags! In *Proceedings of the CoNLL 2017 Shared Task*. pages 207–217.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford Typed Dependencies Representation. In *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*. pages 1–8.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of ICLR*.

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*. 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 ACL-IJCNLP*. pages 334–343.

Jason M. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of COLING*. pages 340–345.

Murhaf Fares, Stephan Oepen, Lilja vrelid, Jari Björne, and Richard Johansson. 2018. The 2018 Shared Task on Extrinsic Parser Evaluation. On the downstream utility of English universal dependency parsers. In *Proceedings of the CoNLL 2018 Shared Task*. page to appear.

Michel Galley and Christopher D. Manning. 2009. Quadratic-Time Dependency Parsing for Machine Translation. In *Proceedings of ACL-IJCNLP*. pages 773–781.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 Shared Task - Automatically Annotated Raw Texts and Word Embeddings. http://hdl.handle.net/11234/1-1989.

Yoav Goldberg. 2016. A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research* 57:345–420.

Carlos Gómez-Rodríguez, Iago Alonso-Alonso, and David Vilares. 2017. How important is syntactic parsing accuracy? an empirical evaluation on rule-based sentiment analysis. *Artificial Intelligence Review* .

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of EMNLP*. pages 1923–1933.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental Joint POS Tagging and Dependency Parsing in Chinese. In *Proceedings of IJCNLP*. pages 1216–1224.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of ACL-IJCNLP*. pages 1681–1691.

Richard Johansson. 2017. EPE 2017: The Trento–Gothenburg opinion extraction system. In *Proceedings of the EPE 2017 Shared Task*. pages 31–39.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of ACL* 4:313–327.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies, 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 EMNLP*. pages 1744–1753.

Emanuele Lapponi, Stephan Oepen, and Lilja Øvrelid. 2017. EPE 2017: The Sherlock negation resolution downstream application. In *Proceedings of the EPE 2017 Shared Task*. pages 25–30.

John Lee, Jason Naradowsky, and David A. Smith. 2011. A Discriminative Model for Joint Morphological Disambiguation and Dependency Parsing. In *Proceedings of ACL-HLT*. pages 885–894.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint Models for Chinese POS Tagging and Dependency Parsing. In *Proceedings of EMNLP*. pages 1180–1191.

Xuezhe Ma and Eduard Hovy. 2017. Neural Probabilistic Model for Non-projective MST Parsing. In *Proceedings of IJCNLP*. pages 59–69.

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.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*. pages 91–98.

Ryan McDonald and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of EACL*. pages 81–88.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980* .

Dat Quoc Nguyen, Mark Dras, and Mark Johnson. 2017. A Novel Neural Network Model for Joint POS Tagging and Graph-based Dependency Parsing. In *Proceedings of the CoNLL 2017 Shared Task*. pages 134–142.

Dat Quoc Nguyen and Karin Verspoor. 2018. From POS tagging to dependency parsing for biomedical event extraction. *arXiv preprint arXiv:1808.03731* .

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*.

Joakim Nivre, Mitchell Abrams, et al. 2018. Universal Dependencies 2.2. http://hdl.handle.net/11234/1-2837.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(2):95–135.

Stephan Oepen, Lilja Øvrelid, Jari Björne, Richard Johansson, Emanuele Lapponi, Filip Ginter, and Erik Velldal. 2017. The 2017 Shared Task on Extrinsic Parser Evaluation. Towards a reusable community infrastructure. In *Proceedings of the EPE 2017 Shared Task*. pages 1–16.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of EMNLP*. pages 1532–1543.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of ACL*. pages 412–418.

Martin Potthast, Tim Gollub, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2014. Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling. In *Proceedings of CLEF Initiative*. pages 268–299.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of ACL* 4:127–141.

Sebastian Schuster, Eric De La Clergerie, Marie Candito, Benoit Sagot, Christopher D. Manning, and Djamé Seddah. 2017. Paris and Stanford at EPE 2017: Downstream Evaluation of Graph-based Dependency Representations. In *Proceedings of the EPE 2017 Shared Task*. pages 47–59.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task*. pages 88–99.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005. Syntax Annotation for the GENIA Corpus. In *Proceedings of IJCNLP: Companion Volume*. pages 220–225.

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 NAACL-HLT*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of ACL-IJCNLP*. pages 323–333.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings IWPT*. pages 195–206.

Liner Yang, Meishan Zhang, Yang Liu, Maosong Sun, Nan Yu, and Guohong Fu. 2018. Joint POS Tagging and Dependence Parsing With Transition-Based Neural Networks. *IEEE/ACM Trans. Audio, Speech & Language Processing* 26(8):1352–1358.

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*. pages 1–20.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, et al. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task*. pages 1–19.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency Parsing as Head Selection. In *Proceedings of EACL*. pages 665–676.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, pos tagging and dependency parsing. In *Proceedings of NAACL-HLT*. pages 42–52.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved Representation Learning for Syntax. In *Proceedings of ACL*. pages 1557–1566.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings ACL-HLT*. pages 188–193.