UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task

Milan Straka
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
straka@ufal.mff.cuni.cz

Abstract

UDPipe is a trainable pipeline which performs sentence segmentation, tokenization, POS tagging, lemmatization and dependency parsing (Straka et al., 2016). We present a prototype for UDPipe 2.0 and evaluate it in the *CoNLL 2018 UD Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, which employs three metrics for submission ranking. Out of 26 participants, the prototype placed first in the MLAS ranking, third in the LAS ranking and third in the BLEX ranking. In extrinsic parser evaluation EPE 2018, the system ranked first in the overall score.

The prototype utilizes an artificial neural network with a single joint model for POS tagging, lemmatization and dependency parsing, and is trained only using the CoNLL-U training data and pretrained word embeddings, contrary to both systems surpassing the prototype in the LAS and BLEX ranking in the shared task.

The open-source code of the prototype is available at `http://github.com/CoNLL-UD-2018/UDPipe-Future`.

After the shared task, we slightly refined the model architecture, resulting in better performance both in the intrinsic evaluation (corresponding to first, second and second rank in MLAS, LAS and BLEX shared task metrics) and the extrinsic evaluation. The improved models will be available shortly in UDPipe at `http://ufal.mff.cuni.cz/udpipe`.

1 Introduction

The Universal Dependencies project (Nivre et al., 2016) seeks to develop cross-linguistically consistent treebank annotation of morphology and syntax for many languages. The latest version of UD 2.2 (Nivre et al., 2018) consists of 122 dependency treebanks in 71 languages. As such, the UD project represents an excellent data source for developing multi-lingual NLP tools which perform sentence segmentation, tokenization, POS tagging, lemmatization and dependency tree parsing.

The goal of the *CoNLL 2018 Shared Tasks: Multilingual Parsing from Raw Text to Universal Dependencies* (*CoNLL 2018 UD Shared Task*) is to stimulate research in multi-lingual dependency parsers which process raw text only. The overview of the task and the results are presented in Zeman et al. (2018). The current shared task is a reiteration of previous year's *CoNLL 2017 UD Shared Task* (Zeman et al., 2017).

This paper describes our contribution to *CoNLL 2018 UD Shared Task*, a prototype of UDPipe 2.0. UDPipe (Straka et al., 2016)[1] is an open-source tool which automatically generates sentence segmentation, tokenization, POS tagging, lemmatization and dependency trees, using UD treebanks as training data. The current version UDPipe 1.2 (Straka and Straková, 2017) is used as a baseline in *CoNLL 2018 UD Shared Task*. UDPipe 1.2 achieves low running times and moderately sized models, however, its performance is behind the current state-of-the-art, placing 13th, 17th and 18th in the three metrics (MLAS, LAS and BLEX, respectively). As a participation system in the shared task, we therefore propose a prototype for UDPipe 2.0, with the goal of reaching state-of-the-art performance.

[1] `http://ufal.mff.cuni.cz/udpipe`

The contributions of this paper are:

- Description of UDPipe 2.0 prototype, which placed 1st in MLAS, 3rd in LAS and 3rd in BLEX, the three metrics of *CoNLL 2018 UD Shared Task*. In extrinsic parser evaluation EPE 2018, the prototype ranked first in overall score.

  The prototype employs an artificial neural network with a single joint model for POS tagging, lemmatization, and parsing. It utilizes solely CoNLL-U training data and word embeddings, and does not require treebank-specific hyperparameter tuning.
- Runtime performance measurements of the prototype, using both CPU-only and GPU environments.
- Ablation experiments showing the effect of word embeddings, regularization techniques and various joint model architectures.
- Post-shared-task model refinement, improving both the intrinsic evaluation (corresponding to 1st, 2nd and 2nd rank in MLAS, LAS and BLEX shared task metrics) and the extrinsic evaluation. The improved models will be available soon in UDPipe.[2]

2 Related Work

Deep neural networks have recently achieved remarkable results in many areas of machine learning. In NLP, end-to-end approaches were initially explored by Collobert et al. (2011). With a practical method for pretraining word embeddings (Mikolov et al., 2013) and routine utilization of recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), deep neural networks achieved state-of-the-art results in many NLP areas like POS tagging (Ling et al., 2015), named entity recognition (Yang et al., 2016) or machine translation (Vaswani et al., 2017).

The wave of neural network parsers was started recently by Chen and Manning (2014) who presented fast and accurate transition-based parser. Many other parser models followed, employing various techniques like stack LSTM (Dyer et al., 2015), global normalization (Andor et al., 2016), biaffine attention (Dozat and Manning, 2016) or recurrent neural network grammars (Kuncoro et al., 2016), improving LAS score in English and Chinese dependency parsing by more than 2 points in 2016. The neural graph-based parser of Dozat et al. (2017) won the last year's *CoNLL 2017 UD Shared Task* by a wide margin.

3 Model Overview

The objective of the shared task is to parse raw texts. In accordance with the CoNLL-U format, the participant systems are required to:

- tokenize the given text and segment it into sentences;
- split multi-word tokens into individual words (CoNLL-U format distinguishes between the surface tokens, e.g., `won't`, and words, e.g., `will` and `not`);
- perform POS tagging, producing UPOS (universal POS) tags, XPOS (language-specific POS) tags and UFeats (universal morphological features);
- perform lemmatization;
- finally perform dependency parsing, including universal dependency relation labels.

We decided to reuse the tokenization, sentence segmentation and multi-word token splitting available in UDPipe 1.2, i.e., the baseline solution, and focus on POS tagging, lemmatization, and parsing, utilizing a deep neural network architecture.

For practical reasons, we decided to devise a joint model for POS tagging, lemmatization, and parsing, with the goal of sharing at least the trained word embeddings, which are usually the largest part of a trained neural network model.

For POS tagging, we applied a straightforward model in the lines of Ling et al. (2015) – first representing each word with its embedding, contextualizing them with bidirectional RNNs (Graves and Schmidhuber, 2005), and finally using a softmax classifier to predict the tags. To predict all three kinds of tags (UPOS, XPOS and UFeats), we reuse the embeddings and the RNNs, and only employ three different classifiers, each for one kind of the tags.

To accomplish lemmatization, we convert each lemma to a rule generating it from the word form, and then classify each input word into one of such rules. Assuming that lemmatization and POS tagging could benefit one another, we reuse the contextualized embeddings of the tagger, and lemmatize through the means of a fourth classifier (in addition to the three classifiers producing UPOS, XPOS and UFeats tags).

[2]http://ufal.mff.cuni.cz/udpipe

Regarding the dependency parsing, we reimplemented a biaffine attention parser of (Dozat et al., 2017), which won the previous year's shared task. The parser also processes contextualized embeddings, followed by additional attention and classification layers. We considered two levels of sharing:

- *loosely joint model*, where only the word embeddings are shared;
- *tightly joint model*, where the contextualized embeddings are shared by the tagger and the parser.

4 Model Implementation

We now describe each model component in a greater detail.

4.1 Tokenization and Sentence Segmentation

We perform tokenization, sentence segmentation and multi-word token splitting with the baseline UDPipe 1.2 approach. In a nutshell, input characters are first embedded using trained embeddings, then fixed size input segments (of 50 characters) are processed by a bidirectional GRU (Cho et al., 2014), and each character is classified into three classes – a) there is a sentence break after this character, b) there is a token break after this character, and c) there is no break after this character. For detailed description, see Straka and Straková (2017).

We only slightly modified the baseline models in the following way: in addition to the segments of size 50 we also consider longer segments of 200 characters during training (and choose the best model for each language according to the development set performance). Longer segments improve sentence segmentation performance for treebanks with nontrivial sentence breaks – such sentence breaks are caused either by the fact that a treebank does not contain punctuation, or that semantic sentence breaks (e.g., end of heading and start of a text paragraph) are not annotated in the treebank. The evaluation of longer segments models is presented later in Section 6.1.

4.2 Embedding Input Words

We represent each input word using three kinds of embeddings, as illustrated in Figure 1.

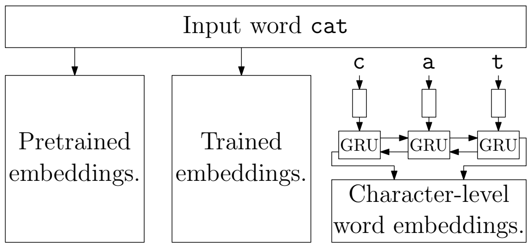

Figure 1: Word embeddings used in the model.

- *pretrained word embeddings*: pretrained word embeddings are computed using large plain texts and are constant throughout the training. We utilize either word embeddings provided by the *CoNLL 2017 UD Shared Task* organizers (of dimension 100), Wikipedia fastText embeddings[3] (of dimension 300), or no pretrained word embeddings, choosing the alternative resulting in highest development accuracy. To limit the size of the pretrained embeddings, we keep at most 1M most frequent words of fastText embeddings, or at most 3M most frequent words of the shared task embeddings.
- *trained word embeddings*: trained word embeddings are created for every training word, initialized randomly, and trained with the rest of the network.
- *character-level word embeddings*: character-level word embeddings are computed similarly as in Ling et al. (2015), utilizing a bidirectional GRU.

4.3 POS Tagging

We process the embedded words through a multi-layer bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to obtain contextualized embeddings. In case multiple RNN layers are employed, we utilize residual connections on all but the first layer (Wu et al., 2016).

For each of the three kinds of tags (UPOS, XPOS and UFeats), we construct a dictionary containing all unique tags from the training data. Then, we employ a softmax classifier for each tag kind processing contextualized embeddings and generating a class from the corresponding tag dictionary.

However, a single-layer softmax classifier has only a limited capacity. To allow more non-linear processing for each tag kind, we prepend a dense layer with tanh non-linearity and a residual connection before each softmax classifier.

[3]http://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

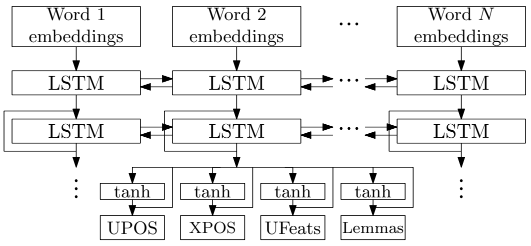

Figure 2: Tagger and lemmatizer model.

The illustration of the tagger and the lemmatizer is illustrated in Figure 2.

4.4 Lemmatization

We lemmatize input words by classifying them into lemma generation rules. We consider these rules as a fourth tag kind (in addition to UPOS, XPOS and UFeats) and use analogous architecture.

To construct the lemma generation rule from a given form and lemma, we proceed as follows:

- We start by finding the longest continuous substring of the form and the lemma. If it is empty, we use the lemma itself as the class.
- If there is a common substring of the form and the lemma, we compute the shortest edit script converting the prefix of the form into the prefix of the lemma, and the shortest edit script converting the suffix of the form to the suffix of the lemma.

  We consider two variants of the edit scripts. The first one permits only character operations `delete_current_char` and `insert_char(c)`. The second variant additionally allows `copy_current_char` operation. For each treebank, we choose the variant producing less unique classes for all training data.
- All above operations are performed case insensitively. To indicate correct casing of the lemma, we consider the lemma to be a concatenation of segments, where each segment is composed of either a sequence of lowercase characters, or a sequence of uppercase characters. We represent the lemma casing by encoding the beginning of every such segment, where the offsets in the first half of the lemma are computed relatively to the start of the lemma, and the offsets in the second half of the lemma are computed relatively to the end of the lemma.

	Lemma generation rules
Minimum	6
Q1	199
Median	490
Mean	877
Q3	831
Maximum	8475

Table 1: Statistics of number of lemma generation rules for 73 treebank training sets of the *CoNLL 2018 UD Shared Task*.

Considering all 73 treebank training sets of the *CoNLL 2018 UD Shared Task*, the number of created lemma generation rules according to the above procedure is detailed in Table 1.

4.5 Dependency Parsing

We base our parsing model on a graph-based biaffine attention parser architecture of the last year's shared task winner (Dozat et al., 2017).

The model starts again with contextualized embeddings produced by bidirectional RNNs, with an artificial ROOT word prepended before the beginning of the sentence. The contextualized embeddings are non-linearly mapped into arc-head and arc-dep representation, which are combined using biaffine attention to produce for each word a distribution indicating the probability of all other words being its dependency head. Finally, we produce an arborescence (i.e., directed spanning tree) with maximum probability by utilizing the Chu-Liu/Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967).

To generate labels for dependency arcs, we proceed analogously – we non-linearly map the contextualized embeddings into rel-head and rel-dep and combine them using biaffine attention, producing for every possible dependency edge a probability distribution over dependency labels.

4.6 Joint Model Variants

We consider two variants of a joint tagging and parsing model, illustrated in Figure 3.

- The *tightly joint model* shares the contextualized embeddings between the tagger and the parser. Notably, the shared contextualized embeddings are computed using 2 layers of bidirectional LSTM. Then, both the tagger and the parser employ an additional layer of bidirectional LSTM, resulting in 4 bidirectional RNN layers.

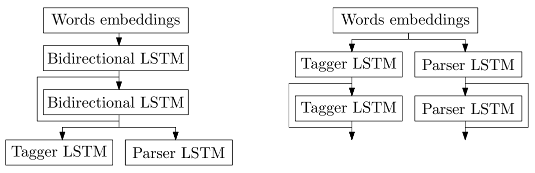

Figure 3: The tightly joint model (on the left) and the loosely joint model (on the right).

- The *loosely joint model* shares only the word embeddings between the tagger and the parser, which both compute contextualized embeddings using 2 layers of bidirectional LSTM, resulting again in 4 RNN layers.

There is one additional difference between the tightly and loosely joint model. While in the tightly joint model the generated POS tags influence the parser model only independently through the shared contextualized embeddings (i.e., the POS tags can be considered regularization of the parser model), the loosely joint model extends the parser word embeddings by the embeddings of the predicted UPOS, XPOS and UFeats tags. Note that we utilize the predicted tags even during training (instead of the gold ones).

4.7 Model Hyperparameters

Considering the 73 treebank training sets of the *CoNLL 2018 UD Shared Task*, we do not employ any treebank-specific hyperparameter search. Most of the hyperparameters were set according to a single Czech-PDT treebank (the largest one), and no effort has been made to adjust them to the other treebanks.

To compute the character-level word embeddings, we utilize character embeddings and GRUs with dimension of 256. The trained word embeddings and the sentence-level LSTMs have a dimension of 512. The UPOS, XPOS and UFeats embeddings, if used, have a dimension of 128. The parser arc-head, arc-dep, rel-head and rel-dep representations have dimensions of 512, 512, 128 and 128, respectively.

4.8 Neural Network Training

For each of the 73 treebanks with a training set we train one model, utilizing only the training treebank and pretrained word embeddings. Each model was trained using the Adam algorithm (Kingma and Ba, 2014) on a GeForce GTX 1080 GPU with a batch size of 32 randomly chosen sentences (for batch size of 64 sentences training ended with out-of-memory error for some treebanks). The training consists of 60 epochs, with the learning rate being 0.001 for the first 40 epochs and 0.0001 for the last 20 epochs. To sufficiently train smaller treebanks, each epoch consists of one pass over the training data or 300 batches, whatever is larger.

Following Dozat and Manning (2016); Vaswani et al. (2017), we modify the default value of β_2 hyperparameter of Adam, but to a different value than both of the above papers – to 0.99, which resulted in best performance on the largest treebank. We also make sure Adam algorithm does not update first and second moment estimates for embeddings not present in a batch.

We regularize the training by several ways:

- We employ dropout with dropout probability 50% on all embeddings and hidden layers, with the exception of RNN states and residual connections.
- We utilize label smoothing of 0.03 in all softmax classifications.
- With a probability of 20%, we replace trained word embedding by an embedding of an unknown word.

With the described training method and regularization techniques, the model does not seem to overfit at all, or very little. Consequently, we do not perform early stopping and always utilize the model after full 60 epochs of training.

The training took 2-4 hours for most of the treebanks, with the two largest Russian-SynTagRus and Czech-PDT taking 15 and 20 hours, respectively.

5 CoNLL 2018 UD Shared Task

The official *CoNLL 2018 UD Shared Task* evaluation was performed using a TIRA platform (Potthast et al., 2014), which provided virtual machines for every participants' systems. During test data evaluation, the machines were disconnected from the internet, and reset after the evaluation finished – this way, the entire test sets were kept private even during the evaluation.

The shared task contains test sets of three kinds:

- For most treebanks large training and development sets were available, in which case we trained the model on the training set and choose among the pretrained word embeddings and tightly or loosely joint model according to performance on the development set.

- For several treebanks very small training sets and no development sets were available. In these cases we manually split 10% of the training set to act as a development set and proceed as in the above case.
- Nine test treebanks contained no training data at all. For these treebanks we adopted the baseline model strategy:

  For Czech-PUD, English-PUD, Finnish-PUD, Japanese-Modern, and Swedish-PUD there were other treebank variants of the same language available in the training set. Consequently, we processed these treebanks using models trained for Czech PDT, English EWT, Finnish TDT, Japanese GSD, and Swedish Talbanken, respectively.

  For Breton-KEB, Faroese-OFT, Naija-NSC, and Thai-PUD, we trained a universal mixed model, by using first 200 sentences of each training set (or less in case of very small treebanks) as training data and first 20 sentences of each development treebank as development data.

5.1 Shared Task Evaluation

The official *CoNLL 2018 UD Shared Task* results are presented in Table 4. In addition to F1 scores, we also include rank of our submission (out of the 26 participant systems).

In the three official metrics (LAS, MLAS and BLEX) our system reached third, first and third average performance. Additionally, our system achieved best average performance in XPOS and AllTags metrics. Furthermore, the lemmatization F1 score was the second best.

Interestingly, although our system achieves highest average score in MLAS (which is a combination of dependency parsing and morphological features), it reaches only third best average LAS and fourth best average UFeats. Furthermore, the `TurkuNLP` participation system surpasses our system in both LAS and UFeats. We hypothesise that the high performance of our system in MLAS metric is caused by the fact that the tagger and parser models are joined, thus producing consistent annotations.

Finally, we note that the segmentation improvements outlined in Section 4.1 resulted in third average F1 score of our system.

Event Extraction	Negation Resolution	Opinion Analysis	Overall Score
49.66 3	58.45 3	60.46 7	**56.19 1**

Table 2: UDPipe 2.0 prototype results in EPE 2018. For each metric we present F1 score percentage and also rank.

	Model size
Average	**139.2MB**
Minimum	90.5MB
Q1	110.0MB
Median	132.0MB
Q3	145.1MB
Maximum	347.1MB
UDPipe 1.2	*13.2MB*

Table 3: Statistics of the model sizes.

5.2 Extrinsic Parser Evaluation

Following the First Shared Task on Extrinsic Parser Evaluation (Oepen et al., 2017), the 2018 edition of Extrinsic Parser Evaluation Initiative (EPE 2018) ran in collaboration with the *CoNLL 2018 UD Shared Task*. The initiative allowed to evaluate the English systems submitted to the CoNLL shared task against three EPE downstream systems – biological event extraction, negation resolution, and fine-grained opinion analysis.

The results of our system are displayed in Table 2. Even though our system ranked only 3rd, 3rd, and 7th in the downstream task F1 scores, it was the best system in the overall score (and average of the three F1 scores).

5.3 Model Size

The statistics of the model sizes is listed in Table 3. Average model size is approximately 140MB, which is more than 10 times larger than baseline UDPipe 1.2 models. Note however that we do not perform any model quantization (which should result in almost four times smaller models, following for example approach of Wu et al. (2016)), and we did not consider the model size during hyperparameter selection. Taking into account that the largest part of the models are the trained word embeddings, the model size could be reduced substantially by reducing trained word embeddings dimension.

5.4 Runtime Performance

The runtime performance of our system is presented in Table 6. Compared to the baseline UDPipe 1.2, tagging and parsing on a single CPU

Language	Tokens	Words	Sentences	UPOS	XPOS	UFeats	AllTags	Lemmas	UAS	LAS	MLAS	BLEX
Afrikaans-AfriBooms	99.75 4	99.75 4	98.25 4	**97.82 1**	**94.23 1**	**97.45 1**	94.08 2	97.11 2	87.74 6	84.99 4	**75.67 1**	75.74 3
Ancient Greek-PROIEL	**100.00 1**	**100.00 1**	49.15 4	97.34 3	97.74 2	**91.98 1**	**90.53 1**	91.08 19	79.50 6	75.78 5	59.82 5	62.62 7
Ancient Greek-Perseus	99.96 4	99.96 4	98.73 3	92.10 4	84.27 2	90.30 2	83.18 2	81.78 20	77.94 7	72.49 6	51.55 6	49.46 7
Arabic-PADT	**99.98 1**	93.71 4	**80.89 1**	90.64 3	87.81 3	88.05 3	87.38 2	88.94 3	76.30 7	72.34 6	63.77 2	65.66 3
Armenian-ArmTDP	97.21 3	96.47 3	92.41 2	65.29 21	—	40.09 23	33.14 21	57.46 20	49.60 3	36.42 2	4.14 20	13.03 8
Basque-BDT	99.97 3	99.97 3	99.83 3	96.08 2	—	92.21 2	**90.13 1**	95.19 4	86.03 5	82.65 5	**71.73 1**	77.25 3
Breton-KEB	91.03 22	90.62 20	90.36 20	36.58 7	**0.01 1**	36.84 4	**0.00 1**	54.99 4	36.54 7	13.87 7	1.24 5	5.78 6
Bulgarian-BTB	99.91 25	99.91 25	95.56 3	98.83 3	**97.01 1**	**97.69 1**	**96.37 1**	97.41 3	92.82 4	89.70 3	**83.12 1**	83.17 2
Buryat-BDT	97.07 3	97.07 3	90.90 3	41.66 8	—	38.34 3	30.00 2	56.83 2	29.20 11	12.61 10	2.09 4	4.41 6
Catalan-AnCora	**99.98 1**	**99.98 1**	**99.68 1**	**98.85 1**	**98.85 1**	**98.36 1**	**97.81 1**	**98.90 1**	92.97 3	90.79 3	**84.07 1**	**85.47 1**
Chinese-GSD	90.04 6	90.04 6	96.25 24	85.83 7	85.64 4	89.13 5	84.61 5	90.01 6	68.73 9	65.67 8	55.97 7	61.38 7
Croatian-SET	**99.93 1**	**99.93 1**	94.64 24	98.02 3	—	**92.12 1**	**91.44 1**	96.69 2	90.33 7	85.93 6	72.85 2	79.84 2
Czech-CAC	**100.00 1**	**99.99 1**	**100.00 1**	99.33 2	**96.36 1**	**96.08 1**	**95.66 1**	98.14 3	92.62 6	90.32 7	**83.42 1**	86.24 4
Czech-FicTree	**100.00 1**	**100.00 1**	98.84 2	98.49 4	94.94 3	95.74 3	94.53 2	97.80 3	93.05 8	90.06 8	81.11 5	84.76 6
Czech-PDT	99.93 4	99.93 3	93.41 2	99.01 3	96.93 2	96.85 2	**96.38 1**	**98.71 1**	92.32 6	90.32 5	**85.10 1**	87.47 3
Czech-PUD	99.25 22	99.25 22	94.95 22	96.84 6	93.64 2	93.55 2	**92.09 1**	**96.44 1**	89.66 5	84.86 4	**75.81 1**	80.32 2
Danish-DDT	**99.90 1**	**99.90 1**	91.41 4	97.72 2	—	97.09 2	96.03 2	96.66 3	85.67 6	83.33 5	75.29 3	76.83 4
Dutch-Alpino	99.93 2	99.93 2	**90.97 1**	**96.90 1**	**94.93 1**	**96.51 1**	**94.36 1**	**96.76 1**	90.11 5	87.09 4	76.09 2	77.76 2
Dutch-LassySmall	99.83 5	99.83 5	76.54 4	95.98 5	94.51 3	95.65 4	93.58 3	95.78 4	86.80 5	83.15 5	72.10 4	72.63 6
English-EWT	99.02 24	99.02 24	77.05 2	95.43 4	95.05 3	95.99 3	93.68 2	**97.23 1**	85.01 8	82.51 7	74.71 3	77.64 4
English-GUM	99.75 2	99.75 2	78.79 4	95.55 5	95.42 2	96.39 3	94.26 2	**96.18 1**	84.60 8	81.35 7	70.74 4	71.67 5
English-LinES	99.93 23	99.93 23	87.21 24	96.77 4	95.72 2	96.67 3	93.26 2	96.44 2	82.73 9	78.26 10	70.66 6	71.73 7
English-PUD	99.67 21	99.67 21	96.19 4	95.68 3	94.59 2	95.32 2	**91.78 1**	95.87 2	87.80 7	85.02 7	74.45 5	78.20 5
Estonian-EDT	99.93 3	99.93 3	91.55 5	97.31 2	98.08 2	**95.81 1**	**94.47 1**	94.88 2	86.29 6	83.26 6	76.62 2	76.29 2
Faroese-OFT	99.50 15	97.40 23	**96.25 1**	62.00 3	**4.23 1**	32.19 7	**1.04 1**	51.31 6	54.95 5	41.99 5	0.74 3	13.49 3
Finnish-FTB	**100.00 1**	**99.99 1**	87.83 3	96.28 3	**95.27 1**	**96.89 1**	93.95 2	94.74 3	88.77 7	86.13 7	78.77 3	78.46 4
Finnish-PUD	99.63 3	99.63 3	90.70 23	97.61 2	—	**96.87 1**	0.00 4	90.64 3	89.69 8	87.69 7	82.17 5	76.00 3
Finnish-TDT	99.68 23	99.68 23	90.32 4	96.75 5	97.52 4	95.43 3	94.17 2	90.18 3	88.10 7	85.72 6	79.25 3	73.99 3
French-GSD	99.69 3	98.81 4	94.29 3	96.32 5	—	96.07 3	95.13 2	96.75 3	88.74 4	85.74 4	77.29 3	79.88 4
French-Sequoia	99.85 2	99.14 4	86.99 4	97.56 6	—	97.00 4	96.25 3	97.36 3	90.07 5	88.04 6	81.45 3	83.04 5
French-Spoken	**100.00 1**	**100.00 1**	21.63 3	95.47 8	97.37 3	—	93.00 7	95.98 3	76.20 7	71.16 7	60.17 6	60.87 6
Galician-CTG	99.89 2	99.21 2	97.23 2	96.98 3	96.62 3	99.05 2	96.24 3	97.53 4	84.42 3	81.88 3	69.75 3	74.46 3
Galician-TreeGal	**99.69 1**	**98.73 1**	83.90 2	94.44 2	**91.64 1**	**93.08 1**	**90.49 1**	95.05 2	78.66 3	**74.25 1**	**60.63 1**	**64.29 1**
German-GSD	99.60 2	99.61 2	82.32 2	94.04 2	96.93 2	89.96 2	84.47 2	96.14 2	82.76 4	78.17 5	56.84 3	69.79 4
Gothic-PROIEL	**100.00 1**	**100.00 1**	29.88 4	95.75 2	96.38 2	**90.05 1**	**87.64 1**	92.39 18	74.92 3	69.39 2	**56.45 1**	61.92 3
Greek-GDT	99.87 2	99.86 3	90.71 4	97.73 3	97.71 3	94.27 3	93.45 2	94.74 4	91.27 3	89.05 3	77.43 2	77.18 5
Hebrew-HTB	99.96 24	85.15 25	99.69 24	82.53 6	82.54 4	81.29 2	80.36 2	82.88 3	67.22 8	63.65 7	51.36 4	54.13 4
Hindi-HDTB	**100.00 1**	**100.00 1**	98.63 24	97.56 2	97.09 2	**94.10 1**	**92.06 1**	98.45 3	94.85 3	91.75 2	**78.30 1**	86.42 3
Hungarian-Szeged	99.79 24	99.79 24	96.12 2	95.48 2	—	92.41 3	91.28 2	92.99 3	83.08 4	78.51 4	**67.13 1**	70.39 4
Indonesian-GSD	**100.00 1**	**100.00 1**	93.53 3	93.55 6	94.45 4	95.77 3	88.78 2	99.60 3	84.91 8	78.58 4	67.58 4	75.94 2
Irish-IDT	99.30 4	99.30 4	92.60 3	91.58 2	90.41 2	82.40 2	78.46 2	87.52 2	78.50 4	70.22 2	45.53 2	51.29 2
Italian-ISDT	99.79 3	99.71 2	**99.38 1**	98.09 2	97.94 2	**97.83 1**	**97.16 1**	**98.21 1**	92.66 4	90.75 4	83.46 3	84.48 3
Italian-PoSTWITA	99.75 2	**99.47 1**	28.95 5	95.99 2	95.77 2	**96.24 1**	94.48 2	94.91 4	77.34 7	73.23 6	61.29 4	61.71 5
Japanese-GSD	90.46 6	90.46 6	**95.01 1**	88.88 7	—	90.45 6	88.88 6	90.01 4	76.13 11	74.54 9	61.99 6	63.48 5
Japanese-Modern	65.98 5	65.98 5	0.00 2	48.51 7	**0.00 1**	64.14 7	**0.00 1**	54.76 6	28.41 16	22.39 15	7.33 15	9.06 15
Kazakh-KTB	93.11 5	92.74 6	81.56 2	48.94 11	49.16 5	46.86 4	38.92 2	57.36 4	39.45 10	24.21 3	7.62 4	9.79 3
Korean-GSD	99.86 4	99.86 4	93.26 3	96.13 3	89.81 4	99.63 3	87.44 4	91.37 3	86.58 6	83.12 6	78.56 6	73.85 3
Korean-Kaist	**100.00 1**	**100.00 1**	**100.00 1**	95.65 2	86.73 3	—	86.62 3	93.53 2	88.13 6	86.16 6	80.46 4	78.15 2
Kurmanji-MG	94.33 3	94.01 3	69.14 3	52.50 22	50.21 20	41.05 21	28.34 18	52.44 21	37.29 5	29.09 3	2.40 19	9.51 12
Latin-ITTB	99.94 4	99.94 4	82.49 3	98.28 5	**95.29 1**	96.36 2	**94.30 1**	98.56 3	88.00 6	85.22 6	79.73 2	82.86 5
Latin-PROIEL	**100.00 1**	**100.00 1**	35.36 5	96.75 2	**96.93 1**	91.26 2	**90.06 1**	95.54 4	73.96 7	69.79 6	58.03 4	64.54 5
Latin-Perseus	99.99 22	99.99 22	**99.15 1**	87.64 7	73.25 2	78.02 4	71.22 2	75.44 20	70.33 6	60.08 7	40.75 4	39.86 7
Latvian-LVTB	99.37 24	99.37 24	**98.66 1**	94.63 6	**87.10 1**	91.48 2	**85.67 1**	93.33 3	83.52 6	79.32 6	67.24 3	70.92 4
Naija-NSC	96.63 7	93.27 6	0.00 4	41.52 21	—	—	5.83 17	89.72 6	26.89 11	12.60 13	3.72 11	11.06 14
North Sami-Giella	99.84 2	99.84 2	98.33 2	90.65 4	91.95 3	86.82 2	81.88 2	78.43 20	74.40 4	68.95 4	54.07 2	48.25 5
Norwegian-Bokmaal	99.81 4	99.81 4	97.13 2	98.14 2	—	96.95 2	**96.20 1**	**98.20 1**	91.81 3	89.98 3	**83.68 1**	**85.82 1**
Norwegian-Nynorsk	99.92 23	99.92 23	93.49 3	97.88 2	—	96.85 2	**95.98 1**	**97.80 1**	91.12 4	88.97 4	**81.86 1**	84.05 3
Norwegian-NynorskLIA	**99.99 1**	**99.99 1**	**99.86 1**	89.74 7	—	89.52 3	84.32 2	92.65 18	67.49 6	59.35 7	46.57 6	49.97 6
Old Church Slavonic-PROIEL	**100.00 1**	**100.00 1**	40.54 4	96.34 2	96.62 2	89.65 2	**88.12 1**	88.93 20	79.06 3	74.84 2	62.60 2	65.71 5
Old French-SRCMF	**100.00 1**	**100.00 1**	**100.00 1**	**96.22 1**	**96.15 1**	**97.79 1**	**95.52 1**	—	**91.72 1**	**87.12 1**	**80.28 1**	**84.11 1**
Persian-Seraji	**100.00 1**	99.65 2	98.75 4	97.32 2	97.33 2	**97.45 1**	**96.90 1**	97.05 2	89.48 4	86.14 3	**80.83 1**	80.28 3
Polish-LFG	99.90 4	99.90 4	99.65 24	98.56 3	**94.18 1**	**95.28 1**	**93.23 1**	96.73 3	96.21 6	94.53 5	**86.93 1**	89.07 3
Polish-SZ	99.99 2	99.88 3	99.00 3	**98.20 1**	92.55 2	92.40 3	**91.36 1**	95.31 3	92.78 6	90.59 6	79.76 5	82.89 3
Portuguese-Bosque	99.64 23	99.51 21	88.47 23	96.37 5	—	95.77 3	93.43 3	97.38 3	89.48 7	87.04 6	74.16 5	80.01 4
Romanian-RRT	99.64 24	99.64 24	95.57 4	97.56 3	96.92 4	97.14 4	96.69 3	97.61 3	90.16 7	85.65 5	77.94 3	79.35 4
Russian-SynTagRus	99.63 3	99.63 3	98.64 4	**98.71 1**	—	**97.17 1**	**96.90 1**	97.94 2	92.96 5	91.46 4	**86.76 1**	87.90 2
Russian-Taiga	**98.14 1**	**98.14 1**	**87.38 1**	90.42 5	**98.12 1**	80.89 4	78.02 4	83.55 4	70.54 8	63.80 8	44.93 5	48.51 6
Serbian-SET	**99.97 1**	**99.97 1**	93.26 2	**98.18 1**	—	**94.26 1**	**93.77 1**	96.56 2	91.68 5	88.15 4	**77.73 1**	81.75 2
Slovak-SNK	**100.00 1**	**100.00 1**	84.96 4	96.65 3	85.98 2	90.33 3	84.60 2	95.66 2	87.96 7	85.06 7	72.08 3	78.44 2
Slovenian-SSJ	98.26 24	98.26 24	76.74 5	96.83 6	93.26 3	93.48 3	92.67 2	96.22 2	87.43 8	85.59 8	77.95 3	81.22 4
Slovenian-SST	**100.00 1**	**100.00 1**	22.90 3	92.58 8	84.68 3	84.65 5	81.57 3	92.56 3	58.32 8	52.84 8	40.24 5	44.57 6
Spanish-AnCora	99.96 24	99.94 22	**98.96 1**	**98.80 1**	**98.80 1**	**98.43 1**	**97.82 1**	**99.02 1**	91.64 7	89.55 6	83.16 3	84.44 3
Swedish-LinES	99.96 2	99.96 2	85.25 3	96.66 4	94.60 4	89.42 4	86.39 3	96.61 2	84.82 8	80.68 8	65.77 6	75.67 4
Swedish-PUD	98.26 24	98.26 24	88.89 23	93.31 6	91.66 3	77.80 5	75.61 3	86.23 6	81.62 8	77.90 8	49.90 4	64.04 4
Swedish-Talbanken	99.88 5	99.88 5	95.79 3	97.79 2	96.43 3	96.59 4	95.43 2	97.08 3	89.32 3	86.36 3	79.08 2	80.73 4
Thai-PUD	8.53 20	8.53 20	0.20 21	5.67 18	0.12 2	6.59 6	0.12 2	—	0.88 9	0.65 13	0.04 7	0.23 17
Turkish-IMST	99.86 2	**97.92 1**	97.09 2	93.58 4	92.82 3	91.25 3	89.00 3	92.74 4	69.34 6	63.07 6	54.02 4	56.69 5
Ukrainian-IU	99.76 2	99.76 2	96.86 2	97.17 3	**91.52 1**	**91.45 1**	**90.12 1**	95.94 3	87.11 5	84.06 5	**72.27 1**	77.11 4
Upper Sorbian-UFAL	**98.64 1**	**98.64 1**	67.24 22	65.51 21	—	49.63 18	43.46 15	63.54 17	35.72 15	24.29 15	3.41 19	11.88 12
Urdu-UDTB	**100.00 1**	**100.00 1**	98.59 24	93.68 6	91.69 5	81.97 6	77.27 4	97.33 2	87.17 8	81.32 7	54.72 6	72.58 4
Uyghur-UDT	99.58 5	99.58 5	82.82 4	88.92 6	91.47 4	86.80 4	78.33 3	92.86 4	76.61 2	65.23 2	**45.78 1**	54.17 2
Vietnamese-VTB	85.05 5	85.05 5	93.31 2	77.61 5	75.91 4	84.83 5	75.80 4	84.76 2	50.98 5	46.45 3	40.26 3	42.88 2
Total	97.46 7	97.04 7	83.64 3	89.37 6	**86.67 1**	86.67 4	**80.30 1**	89.32 2	77.90 5	73.11 3	**61.25 1**	64.49 3

Table 4: Official *CoNLL 2018 UD Shared Task* results of the UDPipe 2.0 prototype. For each metric we present F1 score percentage and also rank (out of the 26 participant systems).

Experiment	UPOS	XPOS	UFeats	AllTags	Lemmas	UAS	LAS	MLAS	BLEX
Shared task submission	95.73	94.79	94.11	91.45	95.12	85.28	81.83	71.71	74.67
Baseline tokenization&segmentation	95.69	94.75	94.07	91.41	95.09	85.10	81.65	71.53	74.48
No precomputed word embeddings	95.23	94.19	93.45	90.52	94.82	84.52	80.88	69.98	73.30
Best model on development set	95.74	94.80	94.12	91.47	95.16	85.28	81.83	71.76	74.69
Checkpoint average of 5 last epochs	95.74	94.81	94.12	91.48	95.14	85.30	81.84	71.80	74.69
No label smoothing	95.66	94.70	94.01	91.26	94.98	85.16	81.68	71.31	74.32
Loosely joint model for all treebanks	95.71	94.82	94.13	91.49	95.29	85.14	81.68	71.71	74.72
Tightly joint model for all treebanks	95.75	94.76	94.06	91.40	94.79	85.34	81.88	71.61	74.33
Tagging each class independently	95.56	94.71	94.04	—	95.47	—	—	—	—
Adding connection between character-level embeddings and the lemma classifier.									
Loosely joint model for all treebanks	95.74	94.85	94.21	91.60	95.86	85.17	81.73	71.89	75.37
Tightly joint model for all treebanks	95.80	94.83	94.17	91.57	95.80	85.44	81.99	71.90	75.54
Best joint model for all treebanks	95.78	94.83	94.19	91.57	95.83	85.38	81.95	71.94	75.55

Table 5: Average score of different variants of our system.

Configuration	Tagging&parsing speed	Speedup to 1 thread CPU
PyPI Tensorflow 1.5.1, CPU version		
1 thread	111 w/s	1
2 threads	191 w/s	1.7
4 threads	326 w/s	2.9
8 threads	517 w/s	4.7
16 threads	624 w/s	5.6
PyPI Tensorflow 1.5.1, GPU version GeForce GTX 1080		
1 thread	2320 w/s	20.9
2 threads	2522 w/s	22.7
4 threads	2790 w/s	25.1
UDPipe 1.2, no parallelism available		
1 thread	*1907 w/s*	*17.2*

Table 6: Average runtime performance of our system.

thread is more than 17 times slower. Utilizing 8 CPU threads speeds up the performance of the prototype by a factor of 4.7, which is still more than 3 times slower than the baseline models. Nevertheless, when GPU is employed during tagging and parsing, the runtime speed of our system surpasses baseline models.

We note that runtime performance has not been a priority during hyperparameter model selection. There are many possible trade-offs which would make inference faster, and some of them will presumably decrease the system performance only slightly.

6 Ablation Experiments

All ablation experiment results in this section are performed using test sets of 61 so called "big treebanks", which are treebanks with provided development data, disregarding small treebanks and test treebanks without training data.

6.1 Baseline Sentence Segmentation

The performance of our system using baseline tokenization and segmentation models (cf. Section 4.1) is displayed in Table 5. The effect of achieving better sentence segmentation influences parsing more than tagging, which can handle wrong sentence segmentation more gracefully.

6.2 Pretrained Word Embeddings

Considering that pretrained word embeddings have demonstrated effective similarity extraction from large plain text (Mikolov et al., 2013), they have a potential of substantially increasing tagging and parsing performance. To quantify their effect, we have evaluated models trained without pretrained embeddings, presenting results in Table 5. Depending on the metric, the pretrained word embeddings improve performance by 0.3-1.7 F1 points.

6.3 Regularization Methods

The effect of early stopping, checkpoint averaging of last 5 epochs and label smoothing is shown also in Table 5. While early stopping and checkpoint averaging have little effect on performance, early stopping demonstrate slight improvement of 0.1-0.4 F1 points.

6.4 Tightly vs Loosely Joint Model

The last model variants presented in Table 5 show the effect of always using either the tightly or loosely joint model for all treebanks. In present implementation, loosely joint model accomplishes better tagging accuracy, while deteriorating parsing slightly. The tightly joint model performs slightly worse during tagging and most notably

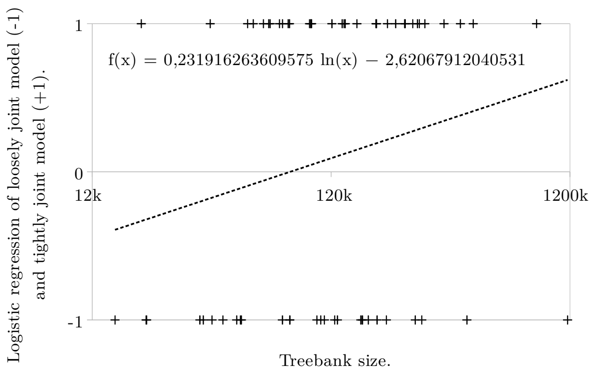

Figure 4: Logarithmic regression of using tightly joint models (+1) and loosely joint models (-1) depending on treebank size in words.

during lemmatization, while improving dependency parsing.

Finally, Figure 4 shows result of logarithmic regression of using tightly or loosely joint model, depending on treebank size in words. Accordingly, the loosely joint model seems to be more suited for smaller treebanks, while the tightly joint model appear to be more suited for larger treebanks.

7 Post-competition Improved Models

Motivated by the decrease in lemmatization performance of the tightly joint model architecture, we refined the architecture of the models by *adding a direct connection from character-level word embeddings to the lemma classifier*. Our hope was to improve lemmatization performance in the tightly joint architecture by providing the lemma classifier a direct access to the embeddings of exact word composition.

As seen in Table 5, the improved models perform considerably better, and with only minor differences between the tighty and loosely joint architectures. We therefore consider only the tightly joint improved models, removing a hyperparameter choice (which joint architecture to use).

The improved models show a considerable increase of 0.68 percentage points in lemmatization performance and minor increase in other tagging scores. The parsing performance also improves by 0.87, 0.19, and 0.16 points in BLEX, MLAS and LAS F1 scores in the ablation experiments.

Encouraged by the results, we performed an evaluation of the improved models in TIRA, achieving 73.28, 61.25, and 65.53 F1 scores in LAS, MLAS and BLEX metrics, which corresponds to increases of 0.17, 0.00 and 1.04 percentage points. Such scores would rank 2nd, 1st, and 2nd in the shared task evaluation. We also submitted the improved models to extrinsic evaluation EPE 2018, improving the F1 scores of the three downstream tasks listed in Table 2 by 0.87, 0.00, and 0.27 percentage points, corresponding to 1st, 3rd, and 4th rank. The overall score of the original models, already the best achieved in EPE 2018, further increased by 0.38 points with the improved models.

8 Conclusions and Future Work

We described a prototype for UDPipe 2.0 and its performance in the *CoNLL 2018 UD Shared Task*, where it achieved 1st, 3rd and 3rd in the three official metrics, MLAS, LAS and BLEX, respectively. The source code of the prototype is available at `http://github.com/CoNLL-UD-2018/UDPipe-Future`.

We also described a minor modification of the prototype architecture, which improves both the intrinsic and the extrinsic evaluation. These improved models will be released shortly in UDPipe at `http://ufal.mff.cuni.cz/udpipe`, utilizing quantization to decrease model size.

Acknowledgments

The work described herein has been supported by the City of Prague under the “OP PPR” program, project No. CZ.07.1.02/0.0/0.0/16_023/0000108, grant GAUK 578218/2018 of Charles University, and it has been using language resources developed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

I would like to thank Jana Straková for considerably improving the quality of the paper.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globaly normalized transition-based neural networks. In *Association for Computational Linguistic*. http://arxiv.org/abs/1603.06042.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

Processing (EMNLP). Association for Computational Linguistics, Doha, Qatar, pages 740–750. http://www.aclweb.org/anthology/D14-1082.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR* abs/1409.1259. http://arxiv.org/abs/1409.1259.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica* 14.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.

Timothy Dozat and Christopher D. Manning. 2016. Deep Biaffine Attention for Neural Dependency Parsing. *CoRR* abs/1611.01734. http://arxiv.org/abs/1611.01734.

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*. Association for Computational Linguistics, Vancouver, Canada, pages 20–30. http://www.aclweb.org/anthology/K17-3002.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 334–343. http://www.aclweb.org/anthology/P15-1033.

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

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* pages 5–6.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2016. What do recurrent neural network grammars learn about syntax? *CoRR* abs/1611.05774. http://arxiv.org/abs/1611.05774.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *CoRR* abs/1508.02096. http://arxiv.org/abs/1508.02096.

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. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association, Portorož, Slovenia, pages 1659–1666.

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. http://hdl.handle.net/11234/1-2837.

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 2017 Shared Task on Extrinsic Parser Evaluation at the Fourth International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*. Pisa, Italy, pages 1 – 16.

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 Evangelos Kanoulas, Mihai Lupu, Paul Clough, Mark Sanderson, Mark Hall, Allan Hanbury, and Elaine Toms, editors, *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14)*. Springer, Berlin Heidelberg New York, pages 268–299. https://doi.org/10.1007/978-3-319-11382-1_22.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the 10th International Conference on Language*

Resources and Evaluation (LREC 2016). European Language Resources Association, Portorož, Slovenia.

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: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 88–99. http://www.aclweb.org/anthology/K/K17/K17-3009.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR* abs/1706.03762. http://arxiv.org/abs/1706.03762.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *CoRR* abs/1603.06270. http://arxiv.org/abs/1603.06270.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Brussels, Belgium, pages 1–20.

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