SParse: Koç University Graph-Based Parsing System for the CoNLL 2018 Shared Task

Berkay Furkan Önder **Can Gümeli** **Deniz Yuret**

Koç University
Artificial Intelligence Laboratory
İstanbul, Turkey
bonder17,cgumeli,dyuret@ku.edu.tr

Abstract

We present SParse, our Graph-Based Parsing model submitted for the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies (Zeman et al., 2018). Our model extends the state-of-the-art biaffine parser (Dozat and Manning, 2016) with a structural meta-learning module, SMeta, that combines local and global label predictions. Our parser has been trained and run on Universal Dependencies datasets (Nivre et al., 2016, 2018) and has 87.48% LAS, 78.63% MLAS, 78.69% BLEX and 81.76% CLAS (Nivre and Fang, 2017) score on the Italian-ISDT dataset and has 72.78% LAS, 59.10% MLAS, 61.38% BLEX and 61.72% CLAS score on the Japanese-GSD dataset in our official submission. All other corpora are evaluated after the submission deadline, for whom we present our unofficial test results.

1 Introduction

End-to-end learning with neural networks has proven to be effective in parsing natural language (Kiperwasser and Goldberg, 2016). Graph-based dependency parsers (McDonald et al., 2005) represent dependency scores between words as a matrix representing a weighted fully connected graph, from which a spanning tree algorithm extracts the best parse tree. This setting is very compatible with neural network models that are good at producing matrices of continuous numbers.

Compared to transition-based parsing (Kırnap et al., 2017; Kiperwasser and Goldberg, 2016), which was the basis of our university's last year entry, graph-based parsers have the disadvantage of producing n^2 entries for parsing an n-word sentence. Furthermore, algorithms used to parse these entries can be even more complex than $O(n^2)$. However, graph-based parsers allow easy-to-parallelize static architectures rather than sequential decision mechanisms and are able to parse non-projective sentences. Non-projective graph-based parsing is the core of last year's winning entry (Dozat et al., 2017).

Neural graph-based parsers can be divided into two components: encoder and decoder. The encoder is responsible for representing the sentence as a sequence of continuous feature vectors. The decoder receives this sequence and produces the parse tree, by first creating a graph representation and then extracting the maximum spanning tree (MST).

We use a bidirectional RNN (bi-RNN) to produce a contextual vector for each word in a sentence. Use of bi-RNNs is the defacto standard in dependency parsing, as it allows representing each word conditioned on the whole sentence. Our main contribution in the encoder part is to the word embeddings feeding the bi-RNN. We use word vectors coming from a language model pretrained on very large language corpora, similar to Kırnap et al. (2017). We extend word embeddings with learnable embeddings for UPOS tags, XPOS tags and FEATs where applicable.

Our decoder can be viewed as a more structured version of the state-of-the-art biaffine decoder of Dozat et al. (2017), where we attempt to condition the label-seeking units to a parse-tree instead of simple local predictions. We propose a meta-learning module that allows structured and unstructured predictions to be combined as a weighted sum. This additional computational complexity is paid off by our simple word-level model in the encoder part. We call it that we call structured meta-biaffine decoder or shortly SMeta.

We implemented our model using Knet deep

Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 216–222
Brussels, Belgium, October 31 – November 1, 2018.
https://doi.org/10.18653/v1/K18-2022

learning framework (Yuret, 2016) in Julia language (Bezanson et al., 2017). Our code will be made available publicly.

We could only get official results for two corpora due to an unexpected software bug. Therefore, we present unofficial results obtained after the submission deadline as well.

2 Related work

Kiperwasser and Goldberg (2016) use trainable BiLSTMs to represent features of each word, instead of defining the features manually. They formulated the structured prediction using hinge loss based on the gold parse tree and parsed scores.

Dozat and Manning (2016) propose deep biaffine attention combined with the parsing model of Kiperwasser and Goldberg (2016), which simplifies the architecture by allowing implementation with a single layer instead of two linear layers.

Stanford's Graph-based Neural Dependency Parser (Dozat et al., 2017) at the CoNLL 2017 Shared Task (Zeman et al., 2017) is implemented with four ReLU layers, two layers for finding heads and dependents of each word, and two layers for finding the dependency relations for each head-dependent pair. The outputs are then fed into two biaffine layers, one for determining the head of the word, and another for determining the dependency relation of head-dependent pair.

We propose a dependency parsing model based on the graph-based parser by Dozat and Manning (2016). We are adding a meta-biaffine decoder layer, similar to the tagging model proposed by Bohnet et al. (2018), for computing the arc labels based on the full tree constructed from the unlabeled arc scores instead of computing them independently.

Our parsing model uses pretrained word embeddings from Kırnap et al. (2017). Our parser uses the same language model with Kırnap et al. (2017), in which graph based-parsing algorithms are applied. However, a transition-based parsing model is given in Kırnap et al. (2017). Therefore, some adaptations are made on the features proposed by Kırnap et al. (2017) in order to use them in a graph based parsing model. We did not use contextual features coming from the language model or features related to words in stack and buffer. Instead, we trained a three-layer BiLSTM from scratch to encode contextual features.

3 Model

In this section, we depict important aspects of our architecture which is shown in Figure 1. We discuss encoder and decoder separately and then give the model hyper-parameters used.

3.1 Encoder

Word Model

We used four main features to represent each word in a sentence: a pre-trained word embedding, UPOS tag embedding, XPOS tag embedding and FEAT embedding.

Pre-trained words come from the language model in Kırnap et al. (2017). This model represents each word using a character-level LSTM, which is a suitable setting for morphologically rich languages, as shown in Dozat et al. (2017). We use the word vectors without further training.

UPOS and XPOS tag embeddings are represented by vectors randomly initialized using unit Gaussian distribution.

Morphological features, also called FEATs, are different in the sense that there are zero or more FEATs for each word. We follow a simple strategy: we represent each FEAT using a randomly initialized vector and add all FEAT embeddings for each word. We simply used zero for word vectors without any morphological features.

For practical reasons, we also needed to represent ROOT word of a sentence. We do so by randomly initializing a word embedding and setting all other embeddings to zero.

At test time, we used tags and morphological features produced by MorphNet (Dayanık et al., 2018). For languages where this model is not available, we directly used UDPipe results (Straka et al., 2016).

Sentence Model

We used a three-layer bidirectional LSTM to represent a sentence. We used the hidden size of 200 for both forward and backward LSTMs. Dropout (Srivastava et al., 2014) is performed at the input of each LSTM layer, including the first layer. Our LSTM simply use nth hidden state for nth word, different from the language model in Kırnap et al. (2017).

The language model discussed in the previous section also provides context embeddings. We performed experiments for combining our own contextual representation with this representation

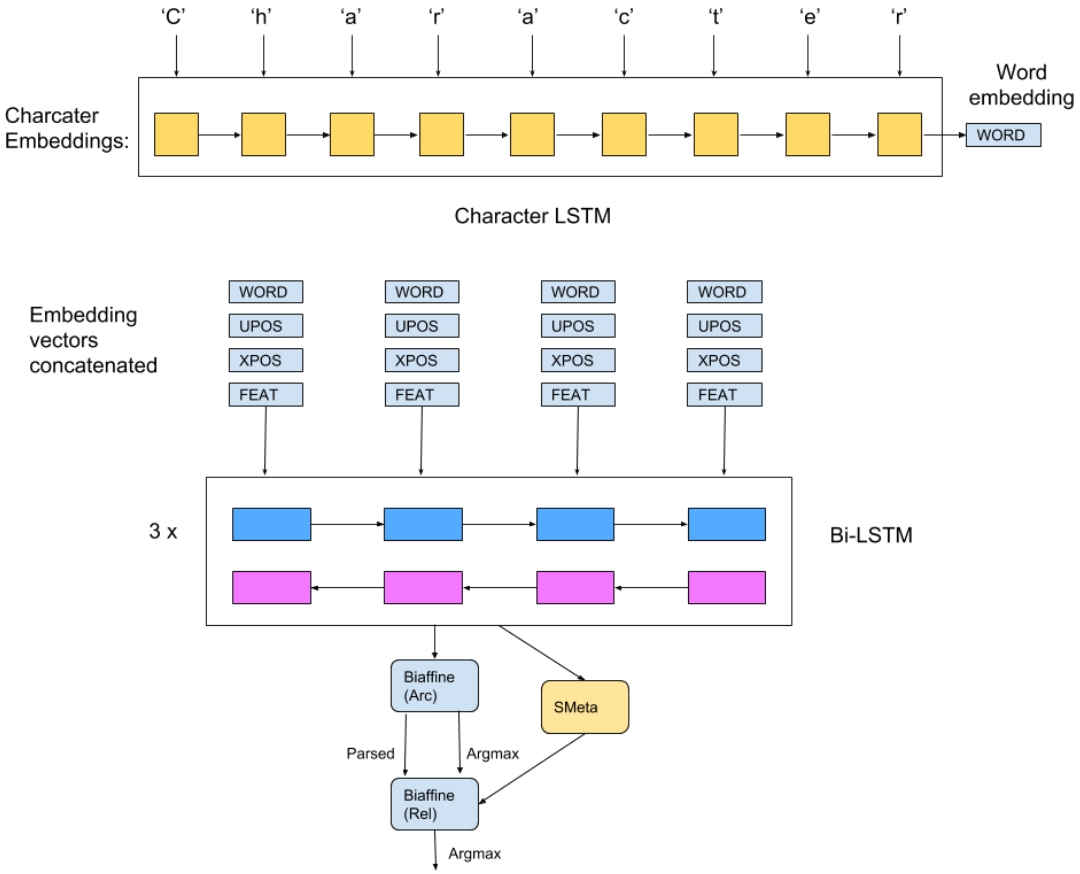

Figure 1: Overall model architecture.

using various concatenation and addition strategies, but we observed poorer performance in terms of generalization. Also, using the language model directly as a feature extractor led to unsatisfactory performance, different from last years' transition-based entry of our institution (Kırnap et al., 2017).

3.2 Decoder

Structured Meta-Biaffine Decoder (SMeta)

Deep biaffine decoder (Dozat and Manning, 2016) is the core of last year's winning entry (Dozat et al., 2017), so we used this module as our starting point. Biaffine architecture is computationally efficient and can be used with easy-to-train softmax objective, different from harder-to-optimize hinge loss objectives as in Kiperwasser and Goldberg (2016).

Similar to (Dozat et al., 2017), we produce four different hidden vectors, two for arcs and two for relations (or labels). Formally

$$\mathbf{h}_i^{(arc-dep)} = \mathbf{MLP}^{(arc-dep)}(\mathbf{h}_i) \quad (1)$$

$$\mathbf{h}_i^{(arc-head)} = \mathbf{MLP}^{(arc-head)}(\mathbf{h}_i) \quad (2)$$

$$\mathbf{h}_i^{(rel-dep)} = \mathbf{MLP}^{(rel-dep)}(\mathbf{h}_i) \quad (3)$$

$$\mathbf{h}_i^{(rel-head)} = \mathbf{MLP}^{(rel-head)}(\mathbf{h}_i) \quad (4)$$

where $\mathbf{h}_i$ represents ith hidden state of the bi-LSTM embedding. The vectors correspond to arcs seeking their dependents, arcs seeking their heads, and corresponding relations. **MLP** can be any neural network module. Here, we simply use dense layers followed by ReLU activations, as in (Dozat et al., 2017).

Now, we perform the biaffine transformation to compute the score matrix representing the graph,

$$\mathbf{s}_i^{(arc)} = H^{(arc-head)} W^{(arc)} \mathbf{h}_i + H^{(arc-head)} \mathbf{b}^{\mathbf{T}(arc)} \quad (5)$$

where $H^{(arc-head)}$ represents matrix of $\mathbf{h}_i^{(arc-head)}$ vectors, $W^{(arc)}$ and $\mathbf{b}^{(arc)}$ are learnable weights.

Up to this point, our decoder is identical to the one in (Dozat et al., 2017). The difference is in the computation of predicted arcs. We compute two different predictions:

$$y_i'^{l(arc)} = \arg\max_j s_{ij}^{(arc)} \tag{6}$$

$$\mathbf{p} = parse(S^{(arc)}) \tag{7}$$

$$y_i'^{s(arc)} = p_i \tag{8}$$

Here $S^{(arc)}$ is the matrix of arc scores and $parse$ is a spanning tree algorithm that computes the indices of the predicted arcs. Now, we compute label scores using these two predictions. First, we compute coefficient vector $\mathbf{k}$ using the bi-RNN encodings,

$$\mathbf{h}' = \frac{\sum_{i=1}^{n} \mathbf{h}_i}{n} \tag{9}$$

$$\mathbf{k} = W^{(meta)}\mathbf{h}' + \mathbf{b}^{(meta)} \tag{10}$$

where n is the number of words in the sentence, W and $\mathbf{b}$ are learned parameters. Averaging over time is inspired by the global average pooling operator in the vision literature (Lin et al., 2013), transforming temporal representation to a global one.

We now compute the weighted sum of label predictions using coefficient vector $\mathbf{k}$.

$$\begin{aligned}\mathbf{s}_i^{l(rel)} = {} & \mathbf{h}_{y_i'^{l(arc)}}^{\mathbf{T}(rel-head)} \mathbf{U}^{(rel)} \mathbf{h}_i^{(rel-dep)} \\ & + W^{(rel)} cat(\mathbf{h}_i^{(rel-dep)}, \mathbf{h}_{y_i'^{l(arc)}}^{(rel-head)}) \\ & + \mathbf{b}^{(rel)}\end{aligned} \tag{11}$$

$$\begin{aligned}\mathbf{s}_i^{s(rel)} = {} & \mathbf{h}_{y_i'^{s(arc)}}^{\mathbf{T}(rel-head)} \mathbf{U}^{(rel)} \mathbf{h}_i^{(rel-dep)} \\ & + W^{(rel)} cat(\mathbf{h}_i^{(rel-dep)}, \mathbf{h}_{y_i'^{s(arc)}}^{(rel-head)}) \\ & + \mathbf{b}^{(rel)}\end{aligned} \tag{12}$$

$$\mathbf{s}_i^{(rel)} = k_1 \mathbf{s}_i^{l(rel)} + k_2 \mathbf{s}_i^{s(rel)} \tag{13}$$

$$y_i'^{(rel)} = \arg\max_j s_{ij}^{(rel)} \tag{14}$$

where $\mathbf{U}^{(rel)}$, $W^{(rel)}$ and $\mathbf{b}^{(rel)}$ are learned parameters.

Our model is trained using sum of softmax losses similar to (Dozat et al., 2017).

Parsing algorithms

In our parsing model, Chu-Liu-Edmonds algorithm (Chu, 1965; Edmonds, 1967) and Eisner (1996)'s algorithm are used interchangeably, during both the training of parser models and parsing phase of test datasets. On the languages whose training dataset consists of more than 250,000 words, Chu-Liu-Edmonds algorithm is used for parsing since it has a complexity of $O(n^2)$, where n is the number of words.

This approach allows us to train our models on relatively larger datasets in less amount of time, compared to the Eisner's algorithm whose time complexity is $O(n^3)$.

On training datasets having at most 250,000 words, Eisner's algorithm is used during both training and parsing phase. Eisners algorithm produces only projective trees and Chu-Liu-Edmonds algorithm produces both projective and non-projective trees. This means the number of possible trees Eisner's algorithm can generate is fewer compared to Chu-Liu-Edmonds algorithm, so even though Eisner's algorithm has higher time complexity than Chu-Liu-Edmonds algorithm, parsing models are trained faster when Eisner's algorithm is used.

3.3 Hyperparameters

We used a 150-dimensional tag and feature embeddings and 350-dimensional word embeddings for the word model. Bi-RNN sentence model has the hidden size of 200 for both forward and backward RNNs, producing 400-dimensional feature context vectors. We used the hidden size of 400 for arc MLPs and 100 for relation MLPs.

4 Training

We used Adam optimizer (Kingma and Ba, 2014) with its standard parameters. Based on dataset size, we trained the model for 25 to 100 epochs and selected the model based on its validation labeled attachment accuracy.

We sampled sentences with identical number of words in a minibatch. In training corpora that are sufficiently large, we sampled minibatches so that approximately 500 tokens exist in a single minibatch. We reduced this size to 250 for relatively small corpora. For very small corpora, we simply sample a constant number of sentences as a minibatch.

Dataset	LAS	MLAS	BLEX	Dataset	LAS	MLAS	BLEX
ar_padt	67.57%	56.22%	58.84%	hu_szeged	73.84%	56.03%	63.31%
bg_btb	85.85%	76.28%	74.73%	id_gsd	76.88%	65.34%	64.89%
ca_ancora	87.60%	79.05%	79.49%	it_isdt	87.51%	78.81%	78.75%
cs_cac	86.05%	73.54%	80.14%	it_postwita	69.48%	56.20%	56.73%
cs_fictree	84.21%	71%	76.86%	ja_modern	22.91%	8.47%	9.73%
cs_pdt	86.07%	76.54%	81.39%	ko_gsd	75.52%	69.15%	63.18%
cs_pud	81.80%	68.19%	75.10%	ko_kaist	81.64%	74.46%	68.86%
cu_proiel	67.80%	56.51%	60.93%	la_ittb	77.66%	68.09%	73.91%
da_ddt	77.90%	68.14%	68.64%	la_perseus	47.45%	29.99%	32.47%
de_gsd	70.65%	34.50%	60.40%	la_proiel	64.29%	51.80%	58.05%
el_gdt	83.65%	66.76%	70.14%	lv_lvtb	71.52%	57.26%	60.20%
en_ewt	77.87%	68.57%	70.83%	nl_alpino	78.28%	63.38%	66.08%
en_gum	75.51%	63.95%	63.52%	nl_lassysmall	78.10%	65.73%	66.77%
en_lines	73.97%	65.15%	66.06%	pl_lfg	88.21%	75.40%	79.25%
en_pud	81.31%	69.85%	73.18%	pl_sz	83.01%	65.10%	73.34%
es_ancora	86.94%	79.14%	79.58%	pt_bosque	84.32%	70.27%	75.29%
et_edt	77.45%	69.58%	66.05%	ro_rrt	82.74%	74.19%	74.60%
eu_bdt	73.84%	60.49%	66.38%	ru_syntagrus	88.22%	80.16%	81.05%
fa_seraji	81.74%	75.15%	71.93%	ru_taiga	40.01%	23.92%	25.44%
fi_ftb	76.62%	66.28%	62.88%	sk_snk	77.81%	56.11%	62.23%
fi_tdt	79.30%	71.07%	64.37%	sl_ssj	78.46%	64.93%	70.50%
fr_gsd	82.32%	72.91%	74.97%	sl_sst	43.63%	31.14%	35.41%
fr_sequoia	82.60%	73.07%	76.07%	sv_lines	74.87%	59.32%	67.25%
fr_spoken	64.62%	53.15%	54.18%	sv_pud	71.66%	43.65%	55.60%
ga_idt	35.84%	13.07%	16.77%	sv_talbanken	79.06%	70.55%	71.16%
gl_ctg	80.31%	67.77%	70.72%	tr_imst	60.19%	49.57%	51.02%
got_proiel	62.40%	49.19%	55.15%	ug_udt	58.48%	38.18%	45.94%
grc_perseus	61.82%	34.17%	41.16%	uk_iu	75.93%	57.81%	64.84%
grc_proiel	68.03%	49.40%	55.62%	ur_udtb	78.41%	51.28%	64.94%
he_htb	60.09%	46.32%	49.19%	vi_vtb	41.27%	34.73%	36.95%
hi_hdtb	87.66%	70%	80.47%	zh_gsd	59.51%	49.33%	54.33%
hr_set	80.86%	60.65%	72.36%				

Table 1: Our unofficial F1 scores. Tests are done in TIRA (Potthast et al., 2014) machine allocated for the task.

We saved best models with corresponding configurations, scores and optimization states during training for recovery. We then re-create the model files for the best models of each corpus by removing the optimization states.

MorphNet is the morphological analysis and disambiguation tool proposed by Dayanık et al. (2018), which we used while training our parsing model. During training of the parser, CoNLL-U formatted training dataset files, which are produced by UDPipe (Straka et al., 2016), are given to MorphNet as input. Then, MorphNet applies its own morphological analysis and disambiguation, and new CoNLL-U formatted files produced by MorphNet are used by our parser.

Even though we used lemmatized and tagged outputs generated by MorphNet while training our parser, we run our parser on outputs generated by UDPipe, due to time constrains during parsing.

5 Results and Discussion

Short after the testing period ended, our parser obtained results on 64 treebank test sets out of 82, which are shown in Table 1. According to the results announced including the unofficial runs, we had an average LAS score of 57% on the 64 test sets on which our model is run and ranked 24th among the best runs of 27 teams. The MLAS score of our model is 46.40% and our model is ranked 22nd out of the submissions of 27 models. And, the BLEX score of our model is 49.17% and our model is ranked 21st out of the best BLEX results of all 27 models including unofficial runs.[1]

According to the results, our model performs better at datasets with comparably larger training data. For instance, our model has around 90% LAS score on Catalan, Indian, Italian, Polish and Russian languages which have higher number of tokens in training data. Furthermore, our model performs relatively well in some low-resource languages like Turkish and Hungarian. However, on the datasets with very small or no training data, such as Japanese Modern, Russian Taiga and Irish IDT, we get lower scores. Hence, our model benefits from large amount of data during training process, but prediction with low resources remains as an issue for our model.

[1]Best results of each team including unofficial runs are announced in http://universaldependencies.org/conll18/results-best-all.html Our results and rankings announced in the paper are taken from the CoNLL 2018 best results webpage in September 2, 2018 and may change with the inclusion of new results of participated teams later.

6 Contributions

In this work, we proposed a new decoding mechanism, called SMeta, for graph-based neural dependency parsing. This architecture attempts to combine structured and unstructured prediction methods using meta-learning. We coupled this architecture with custom training methods and algorithms to evaluate its performance.

Acknowledgments

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) grants 114E628 and 215E201. Special thanks to Ömer Kırnap and Erenay Dayanık for sharing their experiences with us.

References

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. Julia: A fresh approach to numerical computing. *SIAM review* 59(1):65–98.

Bernd Bohnet, Ryan McDonald, Goncalo Simoes, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. *arXiv preprint arXiv:1805.08237* .

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica* 14:1396–1400.

Erenay Dayanık, Ekin Akyürek, and Deniz Yuret. 2018. Morphnet: A sequence-to-sequence model that combines morphological analysis and disambiguation. *arXiv preprint arXiv:1805.07946* .

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734* .

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies* pages 20–30.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B* 71(4):233–240.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 340–345.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *CoRR* abs/1603.04351. http://arxiv.org/abs/1603.04351.

Ömer Kırnap, Berkay Furkan Önder, and Deniz Yuret. 2017. Parsing with context embeddings. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies* pages 80–87.

Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* .

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 523–530.

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, Portoro, Slovenia, pages 1659–1666.

Joakim Nivre and Chiao-Ting Fang. 2017. Universal dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. pages 86–95.

Joakim Nivre et al. 2018. Universal Dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague, http://hdl.handle.net/11234/SUPPLYTHENEWPERMANENTIDHERE!

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.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

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, Portoro, Slovenia.

Deniz Yuret. 2016. Knet: beginning deep learning with 100 lines of julia. In *Machine Learning Systems Workshop at NIPS 2016*.

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.