Proceedings of the First workshop on Resources for African Indigenous Languages (RAIL), pages 25–30
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

Comparing Neural Network Parsers for a Less-resourced and Morphologically-rich Language: Amharic Dependency Parser

Binyam Ephrem Seyoum, Yusuke Miyao, Baye Yimam Mekonnen
Addis Ababa University, University of Tokyo, Addis Ababa University
P.O.Box 1176, Addis Ababa, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, P.O.Box 1176, Addis Ababa
{binyam.ephrem, baye.yimam}@aau.edu.et, yusuke@is.s.u-tokyo.ac.jp

Abstract

In this paper, we compare four state-of-the-art neural network dependency parsers for the Semitic language Amharic. As Amharic is a morphologically-rich and less-resourced language, the out-of-vocabulary (OOV) problem will be higher when we develop data-driven models. This fact limits researchers to develop neural network parsers because the neural network requires large quantities of data to train a model. We empirically evaluate neural network parsers when a small Amharic treebank is used for training. Based on our experiment, we obtain an 83.79 LAS score using the UDPipe system. Better accuracy is achieved when the neural parsing system uses external resources like word embedding. Using such resources, the LAS score for UDPipe improves to 85.26. Our experiment shows that the neural networks can learn dependency relations better from limited data while segmentation and POS tagging require much data.

Keywords: Dependency Parsing, Neural Network, Amharic

1. Introduction

Dependency parsing is the task of analyzing the dependency structure of a given input sentence automatically (Kübler et al., 2009). It requires a series of decisions to form the syntactic structure in the light of dependency relations. Nowadays, dependency grammar is gaining popularity because of its capability to handle predicate-argument structures that are needed in other NLP applications (Mcdonald et al., 2005). In addition, dependency grammar is recommended for languages that have free word order (Kübler et al., 2009; Tsarfaty et al., 2010). However, while dependency parsing is adaptable to many languages, it performs less well with morphologically rich languages like Arabic, Basque, and Greek (Dehdari et al., 2011). It is confirmed in (Habash, 2010), that languages like Arabic, Hebrew, and Amharic present a special challenges for the design of a dependency grammar due to their complex morphology and agreement.

Starting from the mid 20th century, research in NLP has shifted to neural networks. In this line of research, language is represented in the form of non-linear features. The approach is inspired by the the way computation works in the brain (Goldberg, 2017). It is applied in areas such as machine translation, computer vision, and speech recognition. With regards to parsing, the wave of neural network parsers was started in 2014 by Chen and Manning (Chen and Manning, 2014), who presented a fast and accurate transition-based parser using neural networks. Since then other parsing models have employed various techniques such as stack LSTM (Dyer et al., 2015; Kiperwasser and Goldberg, 2016), global normalization (Andor et al., 2016), biaffine attention (Dozat and Manning, 2017) or recurrent neural network grammars (Dyer et al., 2016; Kuncoro et al., 2017). Due to the existence of a treebank for different languages and the shared task of CoNLL 2017 (Zeman et al., 2017) and 2018 (Zeman et al., 2018), large improvements in dependency parsing using neural networks have been reported. For instance, the neural graph-based parser of Dozat et al. (Dozat et al., 2017) won the CoNLL2017 UD Shared Task. In the CoNLL2018 UD Shared Task, the winning system was that of Che et al. (Che et al., 2018). These systems have improved the neural network approach to parsing through the application of optimization functions and external resources such as word embedding. Nowadays, the state-of-the-art in parsing is neural networks incorporating word embedding.

In this paper, we present our experiment on developing a dependency parser for Amharic using the state-of-the-art method. The remaining sections are structured as follows. Section 2 gives a brief background about the process of developing the Amharic treebank and describes the treebank we used for training the neural network models. Section 3 describes the neural parsing systems we use to developed the parser. Section 4 presents our comparison and the results we obtained. The final section, Section 5, summarizes and points out the future directions of the research.

2. Background

A parsing system may use a model which is learned from a treebank to predict the grammatical structure for new sentences. This method of parser development is called data-driven parsing. The goal of data-driven dependency parsing is to learn to accurately predict dependency graphs from the treebank. Following the universal dependency (UD) guidelines, Binyam et al. (Seyoum et al., 2018) developed a treebank for Amharic. In building this resource, they followed a pipeline process. Clitics like prepositions, articles, negation operators, etc. were segmented manually from their host. Then the segmented data were annotated for POS, morphological information and syntactic dependencies based on the UD annotation schema.

The Amharic treebank (ATT) version 1 contains 1,074 manually-annotated sentences (5,245 tokens or 10,010 words). The sentences were collected from grammar books, biographies, news, and fictional and religious texts. The researchers made an effort to include different types of

sentences. The data is included in the UD website[1].

3. Neural Network Parsers

In recent years, many fast and accurate dependency parsers have been made available publicly. Due to the shared task on dependency parsing and the presence of treebanks for different languages, every year new parsing methods have been introduced (Lavelli, 2016; Zeman et al., 2017). Some of the systems require a lot of resources and large treebanks while others are easy to adapt to new languages that have few resources. With this background, we have selected four off-the-shelf parsing systems to test for Amharic. The systems are: UDPipe[2], JPTDP[3], UUParser[4] and Turku[5].

3.1. UDPipe

UDPipe is an open-source and a trainable pipeline parsing system. It performs sentence segmentation, tokenization, part-of-speech tagging, lemmatization, morphological analysis, and dependency parsing (Straka et al., 2016). After a model is trained on the CoNLL-U format, it performs both sentence segmentation and tokenization jointly using a single-layer bidirectional Gated recurrent unit (GRU) network (Straka et al., 2017). The tasks of POS tagging, lemmatization, and morphological analysis, are performed using the MorphoDiTa of (Straková et al., 2014).

The parsing system of UDPipe is based on *Parsito*, (Straková et al., 2014) a transition-based system which is able to parse both non-projective and projective sentences. For non-projective sentences, it employs the arc-standard system of Nivre (Nivre, 2014). To handle non-projective sentences, it has an extra transition called "swap" that reorders two words. It uses neural network classifiers to predict correct transitions. For the purpose of improving parsing accuracy, it adds search-based oracles. It also includes optional beam search decoding, similar to that of Zhang and Nivre (Zhang and Nivre, 2011).

3.2. jPTDP

jPTDP is a joint model for part-of-speech (POS) tagging and dependency parsing (Nguyen and Verspoor, 2018). It was released in two versions; for our experiment, we used the latest version, jPTDP v2.0. This model is based on the BIST graph-based dependency parser of Kiperwasser and Goldberg (Kiperwasser and Goldberg, 2016). Given word tokens in an input sentence, the tagging component uses a BiLSTM to learn latent feature vectors representing the tokens. Then the tagging component feeds these feature vectors into a multi-layer perceptron (MLP) with one hidden layer to predict POS tags.

The parsing component uses another BiLSTM to learn a set of latent feature representations which are based on both the input tokens and the predicted POS tags. These representations are fed to one MLP to decode dependency arcs and another MLP to label the predicted dependency arcs.

[1] https://universaldependencies.org/
[2] https://ufal.mff.cuni.cz/udpipe
[3] https://github.com/datquocnguyen/jPTDP
[4] https://github.com/UppsalaNLP/uuparser
[5] https://turkunlp.org/Turku-neural-parser-pipeline

3.3. UUParser

UUParser (version 2.3) is a pipeline system for dependency parsing that consists of three components (de Lhoneux et al., 2017). The first component performs joint word and sentence segmentation, the second predicts POS tags and morphological features, and the third predicts dependency relation from the words and the POS tags (Nivre, 2008). The word and sentence segmentation is jointly modeled as character-level sequence labeling, employing bidirectional recurrent neural networks (BiRNN) together with CRF (de Lhoneux et al., 2017).

The predictions of POS tags and morphological features are accomplished using a Meta-BiLSTM model with context-sensitive token encoding. This method is adopted from the work of Bohnet et al. (Bohnet et al., 2018). The method applies BiLSTM to modeling both words and characters at the sentence level, giving the model access to the sentence context. The character and word models are combined in the Meta-BiLSTMs. In the Meta-BiLSTM, they concatenate the output, for each word, (of its context sensitive character and word-based embedding) and pass to another BiLSTM to create an additional combined context sensitive encoding. This is followed by a final MLP, whose output is passed onto a linear layer for POS tag prediction.

The third component is dependency parsing, in which a greedy transition-based parser (Nivre, 2008) is applied, following the framework of Kiperwasser and Goldberg (Kiperwasser and Goldberg, 2016). The framework learns representations of tokens in context using BiLSTM. Both the token context and the transition (arc labels) are trained together with a multi-layer perceptron. This enables the model to predict transition and arc labels based on a few BiLSTM vectors. The authors also introduce a static-dynamic oracle, which allows the parser to learn from non-optimal configurations at training time.

3.4. Turku Parser

Turku is a neural parsing pipeline for segmentation, morphological tagging, dependency parsing and lemmatization (Kanerva et al., 2018). For sentence segmentation and tokenization, the system relies on the output of UDPipe. The pipeline allows pre-trained embeddings to be included in the training.

The tagging is done using the system of Dozat et al. (Dozat et al., 2017) which applies a time-distributed affine classifier to the tokens within a sentence. Tokens are first embedded with a word encoder. The encoder sums up a learned token embedding, a pre-trained token embedding, and a token embedding encoded from the sequence of its characters using a unidirectional LSTM. Next, a bidirectional LSTM reads the sequence of embedded tokens in a sentence to create a context-sensitive token representations. These representations are then transformed with ReLU layers separately for each affine tag classification layer (namely UPOS and XPOS). These two classification layers are trained jointly by summing their cross-entropy losses.

Lemmatization is another pipeline in the Turku parser in which the researchers develop their own lemmatization component. The system considers lemmatization as a sequence-to-sequence translation problem. They consider

a word as an input, a sequence of characters which are concatenated with a sequence of its part-of-speech and morphological tags. The output is based on the corresponding lemma represented as a sequence of characters. The researchers essentially train their system to translate the word form characters and morphological tags into lemma characters. They use a deep attention encoder-decoder network with a two-layered bidirectional LSTM encoder for reading the sequence of input characters and morphological tags. As a result, they obtained vectors for the sequence encoder. During the decoding phase, they applied beam search with a size five.

The task of syntactic labeling in the Turku parser is based on the system developed by Dozat et al. (Dozat et al., 2017). The researchers follow methods similar to those that they implemented for the POS tagging module, where tokens were embedded with a word encoder. The word encoder, then, sums up the learned token embedding, a pre-trained token embedding, and a token embedding encoded from the sequence of its characters using unidirectional LSTM. The embedded tokens are then concatenated together with respective POS embeddings. BiLSTM then reads the sequence of embedded tokens in a sentence so that the system has context-aware token representations. The token representations are then transformed using four different ReLU layers separately for two different biaffine classifiers. The classifier scores possible relations (HEAD) and their dependency types (DEPREL), and best predictions are later decoded to form a tree. These relations and type classifiers are again trained jointly by summing up their cross entropy losses. Refer to (Dozat and Manning, 2017) and (Dozat et al., 2017) for the detailed process.

4. Comparing the Neural Network Parsers

Before we discuss the comparison, we describe the experimental set up we followed. The standard practice of preparing data is to divide the data into training, development and test set, usually 80 percent for training, and 10 percent each for development and testing. However, the data set we have is too small to be divided into such proportions. Instead, we carry out ten-fold cross-validation (Zeman et al., 2017), randomly selecting and grouping an equal number of sentences into ten sets. The data we used for this paper are freely available at http://github.com/Binyamephrem/Amharic-treebank. During the training phase, the data in the nine sets are used as a training set and tested against the sentences in the remaining set.

4.1. Experimental Results

In Table 1, we present the results of evaluating the parsing systems we trained. In order to make the evaluation fair, the first experiment is conducted by excluding other external resources. Since the Turku parser requires a pre-trained word embedding, we exclude it from this comparison. For evaluation purposes, we use the conllu18 evaluation script [6], which requires the data to be in the CoNLL-U format and gives us evaluation results for ULA, LAS, MLAS, and BLEX by comparing the system output with the gold data.

Parser	UAS	LAS	MLAS	BLEX
UDPipe	**95.16**	**83.79**	**76.33**	**79.00**
jPTDP	92.42	79.68	69.83	73.83
UUParser	92.00	79.47	70.30	73.66

Table 1: Comparison of the parsing systems

In all measures, UDPipe outperforms both jPTDP and UUParser. LAS computes the percentage of words that are assigned as both the correct syntactic head and the correct dependency label. A system with a higher LAS result will also have a higher result in other measures as well. However, a significant difference is observed in MLAS score (6.03-6.50). MLAS specifically focuses on the combined evaluation of both UPOS and morphological features. Both UDPipe and UUParser are pipeline systems whereas jPTDP is a joint model. The score of jPTDP on MLAS is worse, probably because the model focuses on POS tagging and dependency labeling only. Thus, it may be unjustifiable to compare them on this score as jPTDP does not consider morphological tags in the model. The same logic is applicable regarding the BLEX score. BLEX focuses on the relations between content words by considering lemmas, which are not modeled in jPTDP. The score we obtained for jPTDP probably results from the system seeing gold lemmas or the input data.

4.2. Parsing Model Enhanced with External Resources

We carried out another experiment in which models can be enhanced by external resources. One way of enhancing a model is to use a pre-trained word embedding. For this purpose, we used the trained model for Amharic using fasttext[7]. The data for training the model is from Wikipedia and Common Crawl[8]. The models were trained using continuous bag of words (CBOW) with position-weights, in dimension 300 and considered character of n-grams of length 5 with a window of size 5 and 10 negatives (Grave et al., 2018).

In this comparison, we have included the Turku parser as it requires a pre-trained word embedding. Table 2 presents the results when each model is enhanced with word embedding.

Parser	UAS	LAS	MLAS	BLEX
UDPipe	**96.00**	**85.26**	**77.90**	**80.73**
jPTDP	93.79	82.00	71.42	76.61
UUParser	93.26	79.89	70.65	74.18
Turku	93.26	81.79	68.67	77.36

Table 2: Model enhanced with pre-trained word embedding

We may observe that a pre-trained word embedding increases the performance of the model in each system. The percentage of improvement varies depending the system. jPTPD scores better in all measures, which indicates that the system benefits from the pre-trained model.

[6] http://universaldependencies.org/conll18/conll18_ud_eval.py

[7] https://fasttext.cc/docs/en/crawl-vectors.html

[8] http://commoncrawl.org

For instance, there is a 2.32% improvement over the LAS measure, which could be attributed to the small treebank used in the experiment. Another reason for the greater improvement of the jPTPD model might be related to "unknown" word representation, which are common in morphologically-rich languages. jPTPD uses character-based representations based on LSTM, which produces embeddings from a sequence of characters. This confirms that a character model is better for morphologically-rich languages with high type-token ratios (Smith et al., 2018). Since the UDPipe achieves better results in the first place, the increase due to word embeddings is naturally lower. Such improvements are sometimes evaluated using relative error reduction. For UDPipe, the relative error reduction is [9] approximately $\sim 9\%$. This means that 9% of the errors of the system without word embeddings are reduced using word embedding. Similarly, the relative error reduction for jPTDP is [10] $\sim 11.4\%$. Even by this measure, UDPipe did not benefit as much as jPTDP.

	Plain		segmented	
Metric	**UDPipe**	**Turku**	**UDPipe**	**Turku**
Token	100.00	99.70	100.00	100.00
Sentences	98.62	98.62	100.00	100.00
Words	80.23	80.07	100.00	100.00
UPOS	75.94	77.14	100.00	95.89
XPOS	75.38	76.91	100.00	94.95
UFeats	73.69	74.24	100.00	93.16
AllTags	72.23	74.66	100.00	90.84
Lemmas	80.23	80.07	100.00	100.00
UAS	62.08	61.60	95.16	93.26
LAS	55.32	55.63	83.79	81.79
CLAS	49.33	49.96	78.87	77.36
MLAS	42.74	46.06	76.21	68.67
BLEX	49.33	49.96	78.87	77.36

Table 3: Plain text and segmented text

4.3. Effect of the pipeline on the parsing system

Another experiment we conducted concerns the effect of each pipeline on the performance of the parsers when the input is plain text. For this purpose, we use both UDPipe and the Turku Parser. The remaining parsers need a separate segmentation model or input in CoNLL-U format. We compare the segmentation, tagging and parsing scores of both parsers. Table 3 presents the scores of each system when the input is plain text and segmented text.

We notice that there is a large gap in LAS between the gold and predicted segmentation. This may be caused by poor word predictions, which in turn lowers the tagging prediction. Token and sentence segmentation scores are high for both parsers. However, word segmentation scores for both parsers dropped significantly from 98% to 80%. The tagging result for the Turku parser is better when using gold segmentation (93-95%), but a huge decrease is observed (74-77%) when using predicted segmentation (or plain text). As a result of this, dependency attachment scores also significantly decrease(42â62%). This proves that error propagating in each pipeline greatly affects the attachment scores.

We may also notice from Table 3 that for UDPipe with the segmented input, the system is apparently using the gold POS and morphological features (scores are 100%). Thus, these numbers cannot be compared to the Turku pipeline. For the same reason, LAS scores for Turku with segmented input and UDPipe with segmented input cannot be compared. There is always an improvement when the parser can access gold tags and morphological tags. If there is a perfect tokenizer and tagger, better LAS scores can be obtained. Even though the Turku parser uses the segmentation model from UDPipe, the tagging scores for Turku are slightly better than for UDPipe when plain text is given to both systems.

5. Summary and Future Directions

This paper has presented a comparison of neural network parsers. Based on our comparison, we obtained an LAS score of 83.79 using the UDPipe system. This can be enhanced to 85.26 with external resources, i.e., word embedding. From the experiments we can recommend what will work better for Amharic. A parser for Amharic requires a segmentation of clitics before tagging. For this task, we recommend UDPipe. However, the performance of the segmentation need to be enhanced as it affects the tagging and attachment accuracy greatly.

We have compared both pipeline and joint models for tagging and syntactic parsing. From the pipeline systems, the Turku parsing system achieves better results in the tagging task. However, we have noted that parsing systems that follow the pipeline approach need to have a more efficient segmentation and tagging module. Since errors propagate from one pipeline to another, the parsing or dependency attachment is severely affected. If a joint model is preferred, one needs to consider morphological tagging in addition to POS tagging. Our experiment shows that the joint model is a promising research area for further studies. The jPTDP only focuses on POS and attachment information.

We intend to further our research in two ways. Since the data that the segmentation model was trained on was very limited, we plan to expand the data so that the model for segmentation can be enhanced. We can use the current segmentation prediction on a larger dataset and manually correct the predicted segmentation so as to have a better segmentation model. In addition, we will investigate the effect of learning a joint model of morphological tagging in addition to POS tagging and dependency attachment.

We conclude that, even though we have a small treebank, we can still develop a reasonably efficient parser for Amharic. That is, syntactic patterns can be learned from a small treebank. However, the data in the treebank needs to be carefully selected to include existing syntactic patterns in the language. The challenging aspect in future research will be learning the tags for new lexical items.

6. Bibliographical References

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M. (2016). Globally Normalized Transition-Based Neural Networks. *As-*

[9] It is calculated as (85.26-83.79) / (100-83.79)

[10] It is calculated as (82.00-79.68) / (100-79.68)

sociation for Computational Linguistics, arXiv preprint arXiv:1603.06042.

Bohnet, B., McDonald, R., Simoes, G., Andor, D., Pitler, E., and Maynez, J. (2018). Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 2642–2652, Melbourne, Australia. Association for Computational Linguistics.

Che, W., Liu, Y., Wang, Y., Zheng, B., and Liu, T. (2018). Towards Better UD Parsing: Deep Contextualized Word Embeddings, Ensemble, and Treebank Concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64.

Chen, D. and Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar.

de Lhoneux, M., Shao, Y., Basirat, A., Kiperwasser, E., Stymne, S., Goldberg, Y., and Nivre, J. (2017). From raw text to universal dependencies-look, no tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 207–217.

Dehdari, J., Tounsi, L., and van Genabith, J. (2011). Morphological Features for Parsing Morphologically-rich Languages: A Case of Arabic. In *Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011)*, pages 12–21, Dublin, Ireland. Association for Computational Linguistics.

Dozat, T. and Manning, C. D. (2017). Deep Biaffine Attention for Neural Dependency Parsing. In *ICLR2017*.

Dozat, T., Qi, P., and Manning, C. D. (2017). Stanford's Graph-based Neural Dependency Parser at the CoNLL 2017 Shared Task. (2016):20–30.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (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*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016). Recurrent Neural Network Grammars. In *Proc. of NAACL*.

Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. In *Synthesis Lectures on Human Language Technologies*, volume 10, pages 1–282. Morgan & Claypool Publishers series.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning Word Vectors for 157 Languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, pages 1–5.

Habash, N. Y. (2010). Introduction to Arabic Natural Language Processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

Kanerva, J., Ginter, F., Miekka, N., Leino, A., and Salakoski, T. (2018). Turku Neural Parser Pipeline : An End-to-End System for the CoNLL 2018 Shared Task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 133–142.

Kiperwasser, E. and Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Kübler, S., McDonald, R., and Nivre, J. (2009). Dependency Parsing. *Synthesis Lectures on Human Language Technologies*, 34(1):1–127.

Kuncoro, A., Ballesteros, M., Kong, L., Dyer, C., Neubig, G., and Smith, N. A. (2017). What Do Recurrent Neural Network Grammars Learn About Syntax? *arXiv preprint arXiv:1611.05774*.

Lavelli, A. (2016). Comparing State-of-the-art Dependency Parsers on the Italian Stanford Dependency Treebank. *CLiC it*, pages 173–178.

Mcdonald, R., Crammer, K., and Pereira, F. (2005). Online Large-Margin Training of Dependency Parsers. pages 91–98.

Nguyen, D. Q. and Verspoor, K. (2018). An improved neural network model for joint POS tagging and dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 81–91.

Nivre, J. (2008). Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics*, 34(4):513–553.

Nivre, J. (2014). Universal Dependencies for Swedish. In *The Fifth Swedish Language Technology Conference (Sltc)*, pages 1579–1585.

Seyoum, B. E., Miyao, Y., and Mekonnen, B. Y. (2018). Universal Dependencies for Amharic. In Nicoletta Calzolari, et al., editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 2216–2222, Miyazaki, Japan. European Language Resources Association (ELRA).

Smith, A., de Lhoneux, M., Stymne, S., and Nivre, J. (2018). An Investigation of the Interactions Between Pre-Trained Word Embeddings, Character Models and POS Tags in Dependency Parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720, Brussels, Belgium. Association for Computational Linguistics.

Straka, M., Hajič, J., and Straková, J. (2016). UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In *Proceedings of LREC 2016*, pages 4290–4297.

Straka, M., Straková, J., and Hajič, J. (2017). Prague at EPE 2017: The UDPipe System. 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 Pars-*

ing Technologies, pages 65–74, Pisa (Italy). Association for Computational Linguistics (ACL).

Straková, J., Straka, M., and Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland USA. Association for Computational Linguistics.

Tsarfaty, R., Seddah, D., Goldberg, Y., Kübler, S., Candito, M., Foster, J., Versley, Y., Rehbein, I., and Tounsi, L. (2010). Statistical parsing of morphologically rich languages (SPMRL): what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, Los Angeles, California. Association for Computational Linguistics.

Zeman, D., Popel, M., Straka, M., Hajic, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gokirmak, M., Nedoluzhko, A., Cinkova, S., Hajic jr., J., Hlavacova, J., Kettnerová, V., Uresova, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C. D., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., DePaiva, V., Droganova, K., Martínez Alonso, H., Çöltekin, Ç., Sulubacak, U., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Alcalde, H. F., Strnadová, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonca, G., Lando, T., Nitisaroj, R., and Li, J. (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*, pages 1–19. Association for Computational Linguistics.

Zeman, D., HajiË, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J., and Petrov, S. (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*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

Zhang, Y. and Nivre, J. (2011). Transition-based Dependency Parsing with Rich Non-local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: short papers*, pages 188–193. Association for Computational Linguistics.