Joint Deep Character-Level LSTMs for POS Tagging

KyungTae Lim, Stephen McGregor, Thierry Poibeau

LATTICE - CNRS & École normale supérieure / PSL
Université Sorbonne nouvelle Paris 3 / USPC
1, rue Maurice Arnoux, 92120 Montrouge, France
kyungtae.lim@ens.fr, semcgregor@hotmail.com, thierry.poibeau@ens.fr

Abstract. Character-level feature representation has recently focused on enriching sub-word information by stacking deep neural models. Ideally, applying several character-level representations can help capture different aspects of sub-word information, but this has often failed in the past, mainly because of the nature of the models traditionally used. In this paper, we explore the application of different character-level modeling techniques, presenting a joint training method that separately learns two LSTM-based character representations for a POS tagger. We start by investigating two previously reported techniques, then propose two extended methods: (1) a multi-head attentive character-level representation for capturing several aspects of sub-word information, and (2) an optimal structure for training two different character-level embeddings based on multi-task learning. We evaluate our results on the CoNLL 2018 shared task, showing that our method leads to meaningful improvements for many languages in tagging.

1 Introduction

Natural language processing (NLP) has for long focused on English and a few other languages that were economically (or, more rarely, strategically) profitable. The gradual development of the Web, as well as of social media, has revealed the need to deal with more languages which, in turn, offer new technological challenges. It is, for example, clear that languages exhibit a large diversity of morphological complexity and NLP tools must tackle this diversity in order to obtain acceptable performance beyond English (e.g., on agglutinative or polysynthetic languages).

In this context, character-level word representation is an essential component of NLP tools because of their ability to capture potentially complex morphological information [1, 2]. Let's remind the reader that, traditionally, a character-level word representation learned using Long Short-Term Memory (LSTM) units takes a sequence of characters as input and returns an encoded vector [3]. Recently, studies on character models have focused on enriching feature representations by stacking more neural layers [4], applying an attention mechanism [5], and appending a Multi-Layer Perceptron (MLP) to the output of recurrent networks

[6]. Those approaches have obtained the best performance for part-of-speech (POS) tagging and dependency parsing in the CoNLL 2017 [7] and 2018 [8] shared task. However, despite their benefits, most of these systems also have clear shortcomings, like their (lack of) representation of unknown words. Moreover, the application of several character models, capable of capturing different lexical characteristics, has not been fully explored so far. This is because most of the time when two character models such as CNN and RNN-based character representations are learnt separately, they generally capture almost identical features and thus do not have a real positive influence on the results.

Here we propose a new approach that aims at offering a more accurate representation. This is done through two complementary devices: i) a sub-word analysis, geared to recognize morpheme-like information and ii) a contextual model taking into consideration the sentential framing of a word, which is especially useful for the analysis of unknown words. In order to do this, we need to combine two different character embeddings. One is a context insensitive word-based character representation [4], and the other is a context sensitive sentence-based character representation [9, 10]. We apply joint training to induce two character embeddings focusing on different aspects of sub-word information. Our new technique has the advantage of capturing not only locally optimized character features but also globally optimized features, regardless of language types. We evaluate our system on the CoNLL 2018 shared task data [8], finding statistically-significant improvements compared to the top-performing POS taggers for 10 corpora from 8 languages.

The paper is structured as follows: section 2 describes the two different character embeddings and section 3 describes how they are combined. The experiment and the results are discussed in section 4, leading to the conclusion in section 5. Our tagger and trained models are available in public repositories[1].

2 Two LSTM-based Character Models

In this section, we describe two self-attentive character models. Given an input sentence s of length n with characters s=(ch_1, . , ch_n), our system creates a sequence of sentence-based character vectors $ch^s_{1:n}$, initially encoded as random vectors. Since a sentence s is composed of m words such that s=(w_1, . , w_m), and each word w_i can be decomposed as a sequence of characters w_i=(ch^i_1, . , ch_k), the system also creates a set of sequences of word-based character vectors $ch^{1:m}_{1:k}$. Note that two character embedding, such as $ch^s_{1:n}$ and $ch^{1:m}_{1:k}$, do not refer to the same vector since the system is initialized randomly. A character can thus be represented by two different embeddings.

2.1 Word-based Character Model

State-of-the-art word-based character vectors have been obtained by running a BiLSTM [6] or GRU [11] over the k characters $ch^i_{1:k}$ of a word w_i:

[1] https://github.com/jujbob/Utagger

$$f_j^{(wc)}, b_j^{(wc)} = BiLSTM(r_0^{(wc)}, (ch_1^i, .., ch_k^i))_j$$
$$h_j^{(wc)} = [f_j^{(wc)}; b_j^{(wc)}]$$

Here, $f_j^{(wc)}$ is the forward-pass hidden layer of the BiLSTM for character j of a word, $b_j^{(wc)}$ the backward pass, and $h_j^{(wc)}$ the concatenation of the two. Previous studies have shown that the last encoded character $f_k^{(wc)}$ represents a summary of all the information in an input character sequence [4]. An additional method involves applying the self attention-based linear transformation [5] over the matrix of character encodings $H_i^{(wc)} = h_{1:k}^{(wc)}$, for which attention weights $a_i^{(wc)}$ are calculated as follows[2]:

$$a_i^{(wc)} = Softmax(w^{(wc)} H_i^{(wc)})$$
$$c_i^{(wc)} = a_i^{(wc)} H_i^{(wc)}$$

Here $w^{(wc)}$ is a linear transformation parameter. The self-attention weight $a_i^{(wc)}$ intuitively corresponds to the most informative characters of word w_i for the task being learned. Passing the encoded character vector $H_i^{(wc)}$ of each word through its attention weights $a_i^{(wc)}$, we obtain the character-level word-vector as $c_i^{(wc)}$. Dozat [6] suggest the concatenation of the last encoded vector $f_k^{(wc)}$ and attentive vector $a_i^{(wc)} H_i^{(wc)}$ so as to capture both the summary and sub-word information in one go for tagging. However, as Lin [12] suggest, self-attention based representations tend to focus on a specific component of the sequence. To alleviate this, we propose multi-head attentive character-level word embeddings, which reflect various character-level features of a word by applying multiple attention weights as a matrix $A_i^{(wc)}$ rather than as a vector $a_i^{(wc)}$:

$$A_i^{(wc)} = Softmax(W^{(wc)} \tanh(D^{(wc)} H_i^{(wc)}))$$
$$c_i^{(wc)} = ConcatRow(A_i^{(wc)} H_i^{(wc)}) \quad (1)$$

By applying an attention parameter matrix $W^{(wc)}$, rather than a vector $w^{(wc)}$, and a non-linear function with a weight parameter $D^{(wc)}$, the attention weight can reflect several aspects of sub-word information. For example, a successfully trained attention weight $W^{(wc)}$ could recognize two different morphemes: "de" and "ed" from the word "delexicalized". The effectiveness of the multi-attentive model will be discussed in Section 4.

[2] Here, we use lowercase italics for vectors and uppercase italics for matrices. So a set of hidden state $H_i^{(wc)}$ is a matrix stacked on m characters. In this paper, all the letters w and W denote parameters that the system has to learn. Also, semicolons denote concatenation of two vectors, and $ConcatRow$ denote concatenation of matrix by rows.

2.2 Sentence-based Character Model

The model we have just described is effective at capturing sub-word information, but cannot capture contextual information beyond word boundaries. The overall context can be modeled by encoding the full character sequence from a sentence. Alberti [9] utilized the sentence-based character representation in dependency parsing and achieved state of the art results for morphologically rich languages. In tagging, Bohnet [10] showed that a sentence-based character model that process all characters of a sentence at once is better at keeping context information for unseen data than a token-based one for tagging. For example this model obtained the best results during the 2017 CoNLL shared task. In a similar way, Che [13], who developed the best performing system for the CoNLL 2018 shared task, used sentence-based contextual embeddings with character convolutions and an additional token-based character embedding for dependency parsing.

By extending previous approaches, we create self-attentive sentence-based word embedding, composed of three parts:

1. **Encoding.** A single encoding is generated from the entire character sequence of a sentence using a BiLSTM:

$$h_j^{(sc)} = BiLSTM(r_0^{(sc)}, (ch_1^s, .., ch_n^s))_j$$

2. **Slicing.** The output of a BiLSTM is sliced from the start index *sidx(w$_i$)* to the end index *eidx(w$_i$)* of each word. A matrix $H_i^{(sc)}$ is produced by stacking the encoded character vectors of a word:

$$H_i^{(sc)} = (h_{sidx(w_i)}^{(sc)}, .., h_{eidx(w_i)}^{(sc)})$$

3. **Attention.** $H_i^{(sc)}$ is transformed into a multi-attentive representation $c_i^{(sc)}$, as with the word-based model in (1):

$$A_i^{(sc)} = Softmax(W^{(sc)} \tanh(D^{(sc)} H_i^{(sc)}))$$
$$c_i^{(sc)} = ConcatRow(A_i^{(sc)} H_i^{(sc)})$$

Our approach is distinct from that of Bohnet [10], which proposes the concatenation of only the first and last BiLSTM outputs as $c_i^{(sc)} = MLP([h_{sidx(w_i)}^{(sc)}; h_{eidx(w_i)}^{(sc)}])$. In contrast, we adopt the multi-head attention model proposed in the previous section with $H_i^{(sc)}$, since we believe the multi-attentive model is more accurate to capture the context.

3 Joint Training for a Tagger

Joint Many-Task learning (JMT) enriches context-sensitive feature representations by learning different tasks with shared parameters [14]. This approach can also be applied by training several classifiers for the same task. For example,

meta-Tagger [10] separately trains word-based and character-based POS taggers without parameter sharing and then joins the two models via another meta-BiLSTM and MLP, which is used as the final classifier.

Following Bohnet [10], we train two character-level taggers and then combine them through a meta-BiLSTM tagger. However, we utilize each sentence and word-based character embedding rather than only one ultimate embedding which takes advantage of both the meta-BiLSTM model and JMT by using two character-level models. Separate taggers trained on an individual objective function, each tagger struggles to identify the best features within the limited sentence and word-based character features. We propose applying JMT to generate a shared word embedding between the two character models.

3.1 Two Taggers from Character Models

Our system builds two POS taggers using the two different word embeddings generated from the sentence ($c_i^{(sc)}$) and the word ($c_i^{(wc)}$), as described in Section 2. To enrich word-level contextual information, for each character models, the system concatenates a shared word embedding $w_i^{(e)}$ initialized by a pre-trained word embedding [15] and an ELMo embedding $e_i^{(el)}$ [16] in case that we have, and then passes it through another BiLSTM layer, whose output is $g_i^{(sc)}$ and $g_i^{(wc)}$ for sentence-based and word-based representations respectively. So, for sentence-based embeddings, we apply a two-layer MLP classifier with a weight parameter $Q^{(sc)}$ including a bias term $b^{(sc)}$ to classify the best candidate POS:

$$\begin{aligned} p_i^{(sc)} &= Q^{(sc)} MLP(g_i^{(sc)}) + b^{(sc)} \\ y_i^{(sc)} &= \arg\max_j p_{i,j}^{(sc)} \end{aligned} \tag{2}$$

$$\begin{aligned} p_i^{(wc)} &= Q^{(wc)} MLP(g_i^{(wc)}) + b^{(wc)} \\ y_i^{(wc)} &= \arg\max_j p_{i,j}^{(wc)} \end{aligned} \tag{3}$$

Performing the same operation for the word-based embeddings as well, we predict two POS tags $y_i^{(sc)}$ and $y_i^{(wc)}$.

3.2 Joint POS Tagger

To create joint representations that learn to combine the two taggers' states, we transform the two tagging results as a weighted POS label embedding $h_i^{(pos)}$ as follows:

$$\begin{aligned} l_i^{(sc)} &= \sum_{j=1}^{U} P(y_i^{(sc)} = j | p_i^{(sc)}) pos(j) \\ h_i^{(pos)} &= [l_i^{(sc)}; l_i^{(wc)}] \end{aligned} \tag{4}$$

Here U is the number of possible POS tags, $P(y_i^{(sc)} = j|p_i^{(sc)})$ denotes the probability that the j-th POS tag is assigned to a word w_i, and $pos(j)$ is a randomly initialized POS vector. As Hashimoto [14] suggest, this approach is similar to encoding the k-best POS features. We generate a joint embedding $h_i^{(pos)}$ by concatenating two k-best POS features $[l_i^{(sc)}; l_i^{(wc)}]$. With the classifier we proposed in (2) taking the joint vector $h_i^{(pos)}$ as an input, the system predicts another POS tag $y_i^{(pos)}$ from two k-best POS features. Note that the system only uses tokenized sentences as input, and the k-best POS are predicted at training time.

3.3 Training Three Taggers Simultaneously

Where meta-Tagger [10] trains three taggers using separate optimizations, we trained our taggers simultaneously using a single *Adam*-optimizer, with a summed cross entropy loss for each tagger. This approach has the advantage of passing error propagation directly to the shared word embedding $w^{(e)}$.

4 Experiments and Results

In this section, we present our experimental settings and the results of our model on the data set of the CoNLL 2018 shared task (ST). We compare our results with the official records for this task.[3]

4.1 Data Sets

We evaluate our model on the Universal Dependency 2.2 [17] corpora provided with the ST, following the guideline that the development set is used only for parameter tuning with the provided evaluation metric. In order to compare with the official results, we use permitted pre-trained word embeddings for Japanese and Chinese[4] and other languages [15], and ELMo embeddings trained by Lim [18]. We tested on 10 test treebanks from 8 languages which use different character systems.

As our input test data, we use the word segmentation result of the best performing model for each treebank in order to compare with them directly. Note that when [19] is used for preprocessing, this largely affects the results since tokenization has a direct and massive impact on tagging performance. The segmentation results of the winner have been officially provided by the ST[5] organizers. Among our dataset, five treebanks have no training data available, but for these cases at least one large training treebank in a different domain is available. This offers an excellent opportunity to explore the ability of our joint character model to deal with unseen data. Also, note that ELMo has been

[3] http://universaldependencies.org/conll18/results-upos.html
[4] http://hdl.handle.net/11234/1-1989
[5] http://hdl.handle.net/11234/1-2885

Table 1. universal part-of-speech (UPOS) tagging results compared with the winner (WINN) of each treebank for the ST. Columns denotes the size of training corpus **Size**, and our joint (JOIN), joint with ELMo (JOINE), concatenated (CONC) and ELMo only (ELMO) models. The symbols * and $^+$ represents the winner used ELMo and an ensemble. For ko_gsd, the ko_kaist training corpus is also used.

Corpus	**Size**	WINN	JOINE	ELMO	JOIN	CONC
zh_gsd	3997	91.94*	**93.29**	92.47	91.97	91.81
ja_gsd	7164	92.97*	**93.12**	93.01	92.99	92.83
en_ewt	12543	95.94^{*+}	**95.99**	95.81	95.65	95.39
fr_gsd	14554	96.97	**97.18**	97.04	97.04	96.85
ko_gsd	27410	96.33*	**96.58**	96.15	96.21	96.17
en_pud	0	**96.21**$^{*+}$	96.08	96.17	95.90	95.78
ja_mod	0	54.60	**54.70**	**54.61**	54.67	54.55
cs_pud	0	97.17	-	-	**97.21**	96.81
sv_pud	0	94.28$^+$	-	-	**94.29**	94.09
fi_pud	0	97.65$^+$	-	-	**97.67**	97.50

trained by a sentence-based character model based on external resources. In the end, this gives us the opportunity to investigate three different character models effectively.

4.2 Experimental Setup

We applied the same hyperparameter settings as Smith [20] for BiLSTM dimensions, the MLP, the optimizer including β, and learning rate. We set 300 dimensions for the parameters W and D in (1) and Q in (2). The same dimensionality is applied to the sentence-based character model and the word-based model. In training, we run over the entire training data as an epoch with a batch size of 32 randomly chosen sentences. We save the model with the best performance on the dev set within 300 epochs.

4.3 Results

Table 1 shows the results on the test sets of each treebank, comparing with the best performance WINN as reported for the original ST. The JOIN column represents our model jointly trained by the two different character-level representations described in Section 3.2, and the JOINE column is the JOIN model enhanced with ELMo embeddings as described in Section 3.1.

Overall, our JOINE model achieves state-of-the-art results compared with the official results of the ST winners, except in the case of en_pud, where the best performing model applied both ELMo and an ensemble of different models [18]. Even without the application of ELMo, our JOIN model shows comparable results with models which did use ELMo embeddings (marked *) [13] and an ensemble (marked $^+$) [18].

In Table 1, the last five treebanks have no training data, but there exist large treebanks in different domains for these languages. Here, we can explore the degree to which our joint character model is helpful for handling unseen data. We tested those five treebanks with models trained on other corpora (en_ewt, ja_gsd, cs_pdt, sv_talbanken+sv_lines, and fi_tdt), in line with other approaches to the ST [18, 20]. We can see that our results are comparable for handling cross-domain data.

Impact of the joint learning. To investigate whether our joint model is better than a general concatenation approach, we tested a disjoint model CONC where a word embedding is defined simply as a concatenation of embeddings for different levels of representation:

$$h_i^{(pos)} = [c_i^{(sc)}; c_i^{(wc)}; w_i^{(e)}]$$

It should be noted that the shared word embedding $w^{(e)}$ of our joint learning leads to consistent improvements not only for the joint model but also for the two character-level models. When we use the shared word embedding only for the sentence-based or word-based character models, the performance decreases by an average of 0.05-0.20 absolute points over the three taggers. We can see almost identical experimental results in multi-task learning for training a tagger and parser simultaneously with or without shared embeddings [14].

Impact of ELMo. Although the best performing system [20] for universal part-of-speech (UPOS) tagging in ST outperformed the macro-average score over all the treebanks, with a 0.72 gap over the second-ranked team, some teams who have used ELMo got the best score on many languages. It is thus necessary to extend the evaluation by using the same ELMo embedding, as applied in ST[6]. To investigate the performance of previously proposed taggers with ELMo, we evaluated ELMO model which trained based on concatenation of ELMo and word embedding as:

$$h_i^{(pos)} = [e_i^{(el)}; w_i^{(e)}]$$

In Table 2, ELMO model generally shows better performances than our JOIN model but not for JOINE. We found a marginal gap between ELMO and JOINE in tagging Chinese and Korean than English and French. As reported by [21], we assume that languages have a bigger character set size gain much more influences by character embeddings not only trained on the limited training data but also external data.

While testing ELMo for our JOINE model, we found the concatenation of ELMo embedding for both sentence and word models simultaneously lead to performance degradation because the dimension of ELMo is relatively huge (1,024). To avoid the problem, We did not use ELMo embedding for encoding our word-based character-level embedding $c_i^{(wc)}$ proposed in Section 3.1. There is some indication that in the less-resourced conditions, ELMo is more influential than

[6] https://github.com/jujbob/multilingual-models

Table 2. eu_bdt tagging results by the number of rows N of the word and sentence-based character embedding. Here, WORD and SENT denote models which trained taggers only word and sentence-based character representations (as described in (1))

# of head	N=1	N=2	N=3	N=5
WORD	96.04	96.17	**96.19**	96.12
SENT	96.24	**96.26**	96.21	96.17
JOIN	96.39	96.40	**96.43**	96.35

the dynamically trained character embeddings. This is because ELMo is trained on corpus resources external to the task; in contrast, our character models are trained only on the limited training data, and so struggle to learn deep contexts (see the size and performance gaps between JOIN and JOINE on zh_gsd).

Impact of the self-attentive approach. Table 2 demonstrates the effectiveness of the multi-head attention component of the word and sentence-based character models (as described in (1)). Here, N represents the number of rows allocated to the matrix A_i, with additional columns providing traction for the model to focus on different semantic components of the word being modeled.

We see at least marginal improvements when expanding from a single-head N=1 to double-head N=2 for all models, and then to triple-head N=3 for the WORD and JOIN models. Here, the model applied N=1 is the identical single head model widely used proposed by [6]. We observe a negative impact when expanding beyond 5 rows for all models. This is because, as Lin [12] and Vaswani [22] show, each additional attentive-score $a_i^{(wc)}$ tends to be focused on the same part of a sequence, even though it requires an N-times higher dimensional space.

5 Conclusion

In this paper we have presented a tagging model involving two different character-level components, the first one was based on word boundaries, whereas the second was able to take into account contextual information at the sentence level. By training two individual taggers based on two different character models, we have produced a tagger taking into account not only locally optimized character information but also globally optimized information, regardless of the language types. We have detailed our three main innovations: (*1*) Multi-attentive character model, which leads the system to capture several aspects of sub-word information. (*2*) Joint POS representations to combine the two taggers' states as a feature for final tagger and (*3*) Contextual representation to capture context information from external resources. This method is effective, improving on previously reported results. For future work, we plan to integrate our enhanced morphological tagger with our dependency parser.

References

1. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. In: AAAI. (2016) 2741–2749
2. Yu, X., Vu, N.T.: Character composition model with convolutional neural networks for dependency parsing on morphologically rich languages. CoRR **abs/1705.10814** (2017)
3. Ballesteros, M., Dyer, C., Smith, N.A.: Improved transition-based parsing by modeling characters instead of words with lstms. CoRR **abs/1508.00657** (2015)
4. Shi, T., Wu, F.G., Chen, X., Cheng, Y.: Combining global models for parsing universal dependencies. In: Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, Association for Computational Linguistics (2017) 31–39
5. Cao, K., Rei, M.: A joint model for word embedding and word morphology. CoRR **abs/1606.02601** (2016)
6. Dozat, T., Qi, P., Manning, C.D.: 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, Vancouver, Canada, Association for Computational Linguistics (2017) 20–30
7. Zeman, D., Ginter, F., Hajič, J., Nivre, J., Popel, M., Straka, M., et al: 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 (2017) 1–20
8. Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J., Petrov, S.: 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, Brussels, Belgium, Association for Computational Linguistics (2018) 1–21
9. Alberti, C., Andor, D., Bogatyy, I., Collins, M., Gillick, D., Kong, L., Koo, T., Ma, J., Omernick, M., Petrov, S., Thanapirom, C., Tung, Z., Weiss, D.: Syntaxnet models for the CoNLL 2017 shared task. CoRR **abs/1703.04929** (2017)
10. Bohnet, B., McDonald, R.T., Simões, G., Andor, D., Pitler, E., Maynez, J.: Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. CoRR **abs/1805.08237** (2018)
11. Straka, M.: UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, Association for Computational Linguistics (2018) 197–207
12. Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding. CoRR **abs/1703.03130** (2017)
13. Che, W., Liu, Y., Wang, Y., Zheng, B., Liu, T.: 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, Brussels, Belgium, Association for Computational Linguistics (2018) 55–64
14. Hashimoto, K., Xiong, C., Tsuruoka, Y., Socher, R.: A joint many-task model: Growing a neural network for multiple NLP tasks. CoRR **abs/1611.01587** (2016)
15. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. CoRR **abs/1607.04606** (2016)

16. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. CoRR **abs/1802.05365** (2018)
17. Zeman, D., et al.: Universal Dependencies 2.2 CoNLL 2018 shared task development and test data (2018) LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague, http://hdl.handle.net/11234/1-2184.
18. Lim, K., Park, C., Lee, C., Poibeau, T.: SEx BiST: A multi-source trainable parser with deep contextualized lexical representations. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, Association for Computational Linguistics (2018) 143–152
19. Straka, M., Hajič, J., Straková, J.: 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), Portoro, Slovenia, European Language Resources Association (2016)
20. Smith, A., Bohnet, B., de Lhoneux, M., Nivre, J., Shao, Y., Stymne, S.: 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, Association for Computational Linguistics (2018) 113–123
21. Smith, A., de Lhoneux, M., Stymne, S., Nivre, J.: 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. (2018) 2711–2720
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., eds.: Advances in Neural Information Processing Systems 30. Curran Associates, Inc. (2017) 5998–6008