Proceedings of 1st Workshop on Language Technologies for Historical and Ancient Languages, pages 124–129
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

UDPipe at EvaLatin 2020: Contextualized Embeddings and Treebank Embeddings

Milan Straka, Jana Straková
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{straka,strakova}@ufal.mff.cuni.cz

Abstract

We present our contribution to the EvaLatin shared task, which is the first evaluation campaign devoted to the evaluation of NLP tools for Latin. We submitted a system based on UDPipe 2.0, one of the winners of the CoNLL 2018 Shared Task, The 2018 Shared Task on Extrinsic Parser Evaluation and SIGMORPHON 2019 Shared Task. Our system places first by a wide margin both in lemmatization and POS tagging in the open modality, where additional supervised data is allowed, in which case we utilize all Universal Dependency Latin treebanks. In the closed modality, where only the EvaLatin training data is allowed, our system achieves the best performance in lemmatization and in classical subtask of POS tagging, while reaching second place in cross-genre and cross-time settings. In the ablation experiments, we also evaluate the influence of BERT and XLM-RoBERTa contextualized embeddings, and the treebank encodings of the different flavors of Latin treebanks.

Keywords: EvaLatin, UDPipe, lemmatization, POS tagging, BERT, XLM-RoBERTa

1. Introduction

This paper describes our participant system to the EvaLatin 2020 shared task (Sprugnoli et al., 2020). Given a segmented and tokenized text in CoNLL-U format with surface forms as in

```
# sent_id = 1
1  Dum        _          _       _   ...
2  haec       _          _       _   ...
3  in         _          _       _   ...
4  Hispania   _          _       _   ...
5  geruntur   _          _       _   ...
6  C.         _          _       _   ...
7  Trebonius  _          _       _   ...
```

the task is to infer lemmas and POS tags:

```
# sent-id = 1
1  Dum        dum        SCONJ   _   ...
2  haec       hic        DET     _   ...
3  in         in         ADP     _   ...
4  Hispania   Hispania   PROPN   _   ...
5  geruntur   gero       VERB    _   ...
6  C.         Gaius      PROPN   _   ...
7  Trebonius  Trebonius  PROPN   _   ...
```

The EvaLatin 2020 training data consists of 260k words of annotated texts from five authors. In the closed modality, only the given training data may be used, while in open modality any additional resources can be utilized.

We submitted a system based on UDPipe 2.0 (Straka et al., 2019a). In the open modality, our system also uses all three UD 2.5 (Zeman et al., 2019) Latin treebanks as additional training data and places first by a wide margin both in lemmatization and POS tagging.

In the closed modality, our system achieves the best performance in lemmatization and in classical subtask of POS tagging (consisting of texts of the same five authors as the training data), while reaching second place in cross-genre and cross-time setting.

Additionally, we evaluated the effect of:

- BERT (Devlin et al., 2019) and XLM-RoBERTa (Conneau et al., 2019) contextualized embeddings;
- various granularity levels of treebank embeddings (Stymne et al., 2018).

2. Related Work

The EvaLatin 2020 shared task (Sprugnoli et al., 2020) is reminiscent of the SIGMORPHON2019 Shared Task (McCarthy et al., 2019), where the goal was also to perform lemmatization and POS tagging, but on 107 corpora in 66 languages. It is also related to CoNLL 2017 and 2018 Multilingual Parsing from Raw Texts to Universal Dependencies shared tasks (Zeman et al., 2017; Zeman et al., 2018), in which the goal was to process raw texts into tokenized sentences with POS tags, lemmas, morphological features and dependency trees of the Universal Dependencies project (Nivre et al., 2016), which seeks to develop cross-linguistically consistent treebank annotation of morphology and syntax for many languages.

UDPipe 2.0 (Straka et al., 2016; Straka, 2018) was one of the winning systems of the CoNLL 2018 shared task, performing the POS tagging, lemmatization and dependency parsing jointly. Its modification (Straka et al., 2019a) took part in the SIGMORPHON 2019 shared task, delivering best performance in lemmatization and comparable to best performance in POS tagging.

A new type of deep contextualized word representation was introduced by Peters et al. (2018). The proposed embeddings, called ELMo, were obtained from internal states of deep bidirectional language model, pretrained on a large text corpus. The idea of ELMos was extended to BERT by Devlin et al. (2019), who instead of a bidirectional recurrent language model employ a Transformer (Vaswani

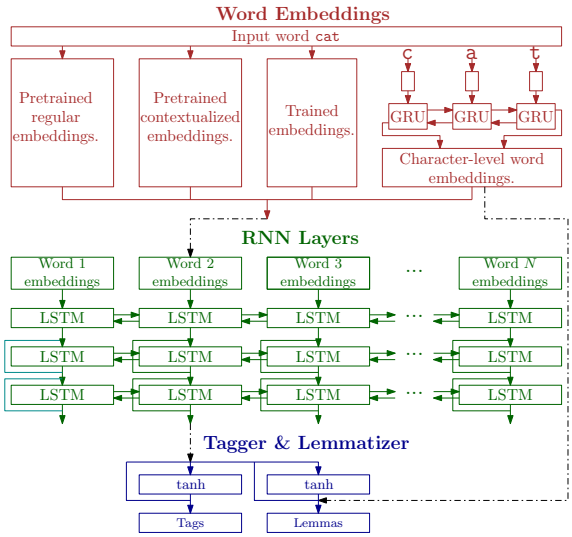

Figure 1: The UDPipe network architecture of the joint tagger and lemmatizer.

et al., 2017) architecture. A multilingual BERT model trained on 102 languages can significantly improve performance in many NLP tasks across many languages. Recently, XLM-RoBERTa, an improved multilingual model based on BERT, was proposed by Conneau et al. (2019), which appears to offer stronger performance in multilingual representation (Conneau et al., 2019; Lewis et al., 2019).

3. Methods

3.1. Architecture Overview

Our architecture is based on UDPipe entry to SIGMORPHON 2019 Shared Task (Straka et al., 2019a), which is available at `https://github.com/ufal/sigmorphon2019`. The resulting model is presented in Figure 1.

In short, the architecture is a multi-task model predicting jointly lemmas and POS tags. After embedding input words, three shared bidirectional LSTM (Hochreiter and Schmidhuber, 1997) layers are performed. Then, softmax classifiers process the output and generate the lemmas and POS tags.

The lemmas are generated by classifying into a set of edit scripts which process input word form and produce lemmas by performing character-level edits on the word prefix and suffix. The lemma classifier additionally takes the character-level word embeddings as input. The lemmatization is further described in Section 3.2.

The input word embeddings are the same as in the previous versions of UDPipe 2.0:

- **end-to-end word embeddings**,
- **character-level word embeddings:** We employ bidirectional GRUs (Cho et al., 2014; Graves and Schmidhuber, 2005) of dimension 256 in line with (Ling et al., 2015): we represent every Unicode character with a vector of dimension 256, and concatenate GRU output for forward and reversed word characters. The character-level word embeddings are trained together with UDPipe network.
- **pretrained word embeddings:** We use FastText word embeddings (Bojanowski et al., 2017) of dimension 300, which we pretrain on plain texts provided by CoNLL 2017 UD Shared Task (Ginter et al., 2017), using segmentation and tokenization trained from the UD data.[1]
- **pretrained contextualized word embeddings:** We use the `Multilingual Base Uncased` BERT (Devlin et al., 2019) model to provide contextualized embeddings of dimensionality 768, averaging the last layer of subwords belonging to the same word.

We refer the readers for detailed description of the architecture and the training procedure to Straka et al. (2019a).

3.2. Lemmatization

The lemmatization is modeled as a multi-class classification, in which the classes are the complete rules which lead from input forms to the lemmas. We call each class encoding a transition from input form to lemma a *lemma rule*. We create a lemma rule by firstly encoding the correct casing as a *casing script* and secondly by creating a sequence of character edits, an *edit script*.

Firstly, we deal with the casing by creating a *casing script*. By default, word form and lemma characters are treated as lowercased. If the lemma however contains upper-cased characters, a rule is added to the casing script to uppercase the corresponding characters in the resulting lemma. For example, the most frequent casing script is "keep the lemma lowercased (don't do anything)" and the second most frequent casing script is "uppercase the first character and keep the rest lowercased".

As a second step, an *edit script* is created to convert input lowercased form to lowercased lemma. To ensure meaningful editing, the form is split to three parts, which are then processed separately: a prefix, a root (stem) and a suffix. The root is discovered by matching the longest substring shared between the form and the lemma; if no matching substring is found (e.g., form *eum* and lemma *is*), we consider the word irregular, do not process it with any edits and directly replace the word form with the lemma. Otherwise, we proceed with the edit scripts, which process the prefix and the suffix separately and keep the root unchanged. The allowed character-wise operations are character copy, addition and deletion.

The resulting *lemma rule* is a concatenation of a casing script and an edit script. The most common lemma rules present in EvaLatin training data are presented in Table 1.

Using the generated lemma rules, the task of lemmatization is then reduced to a multiclass classification task, in which the artificial neural network predicts the correct lemma rule.

3.3. Treebank Embedding

In the open modality, we additionally train on all three UD 2.5 Latin treebanks. In order to recognize and handle possible differences in the treebank annotations, we employ treebank embeddings following (Stymne et al., 2018).

[1] We use `-minCount 5 -epoch 10 -neg 10` options.

Lemma Rule	Casing Script	Edit Script	Most Frequent Examples
↓0;d¦	all lowercase	do nothing	et→et, in→in, non→non, ut→ut, ad→ad
↓0;d¦-+u+s	all lowercase	change last char to us	suo→suus, loco→locus, Romani→romanus, sua→suus
↓0;d¦---+o	all lowercase	change last 3 chars to o	dare→do, dicere→dico, fieri→fio, uidetur→uideo, data→do
↓0;d¦-+s	all lowercase	change last char to s	quid→quis, id→is, rei→res, omnia→omnis, rem→res
↓0;d¦----+o	all lowercase	change last 4 chars to o	hominum→homo, dedit→do, homines→homo
↓0;d¦--+o	all lowercase	change last 2 chars to o	habere→habeo, dicam→dico, ferre→fero, dat→do
↓0;d¦--+u+s	all lowercase	change last 2 chars to us	publicae→publicus, suis→suus, suam→suus, suos→suus
↓0;d¦-	all lowercase	remove last character	gratiam→gratia, causam→causa, uitam→uita, copias→copia
↓0;d¦-+u+m	all lowercase	change last char to um	belli→bellum, posse→possum, bello→bellum
↓0;d¦---+s	all lowercase	change last 3 chars to s	omnibus→omnis, rebus→res, nobis→nos, rerum→res
↑0¦↓1;d¦	1st upper, then lower	do nothing	Caesar→Caesar, Plinius→Plinius, Antonius→Antonius
↓0;d¦-----+o	all lowercase	change last 5 chars to o	uideretur→uideo, uidebatur→uideo, faciendum→facio
↓0;d¦--+i	all lowercase	change last 2 chars to i	quod→qui, quae→qui, quem→qui, quos→qui, quam→qui
↓0;d¦---	all lowercase	remove last 3 characters	quibus→qui, legiones→legio, legionum→legio, legionis→legio
↓0;d¦--+s	all lowercase	change last 2 chars to s	omnium→omnis, hostium→hostis, parte→pars, urbem→urbs
...	...	...	...
↓0;ais	all lowercase	ignore form, use is	eum→is, eo→is, ea→is, eorum→is, eam→is

Table 1: Fifteen most frequent lemma rules in EvaLatin training data ordered from the most frequent one, and the most frequent rule with an absolute edit script.

System	Lemmatization		
	classical	cross-genre	cross-time
UDPipe – open	96.19 (1)	87.13 (1)	91.01 (1)
UDPipe – closed	95.90 (2)	85.47 (3)	87.69 (2)
P2 – closed 1	94.76 (3)	85.49 (2)	85.75 (3)
P3 – closed 1	94.60 (4)	81.69 (5)	83.92 (4)
P2 – closed 2	94.22 (5)	82.69 (4)	83.76 (5)
Post ST – open	*96.35*	*87.48*	*91.07*
Post ST – closed	*95.93*	*85.94*	*87.88*

Table 2: Official ranking of EvaLatin lemmatization. Additionally, we include our best post-competition model in italic.

System	Tagging		
	classical	cross-genre	cross-time
UDPipe – open	96.74 (1)	91.11 (1)	87.69 (1)
UDPipe – closed	96.65 (2)	90.15 (3)	84.93 (3)
P4 – closed 2	96.34 (3)	90.64 (2)	87.00 (2)
P3 – closed 1	95.52 (4)	88.54 (4)	83.96 (4)
P4 – closed 3	95.35 (5)	86.95 (6)	81.38 (7)
P2 – closed 1	94.15 (6)	88.40 (5)	82.62 (6)
P4 – closed 1	93.24 (7)	83.88 (7)	82.99 (5)
P2 – closed 2	92.98 (8)	82.93 (8)	80.78 (8)
P5 – closed 1	90.65 (9)	73.47 (9)	76.62 (9)
Post ST – open	*96.82*	*91.46*	*87.91*
Post ST – closed	*96.76*	*90.50*	*84.70*

Table 3: Official ranking of EvaLatin lemmatization. Additionally, we include our best post-competition model in italic.

Furthermore, given that the author name is a known information both during training and prediction time, we train a second model with author-specific embeddings for the individual authors. We employ the model with author-specific embeddings whenever the predicted text comes from one of the training data authors (in-domain setting) and a generic model otherwise (out-of-domain setting).

4. Results

The official overall results are presented in Table 2 for lemmatization and in Table 3 for POS tagging. In the open modality, our system places first by a wide margin both in lemmatization and POS tagging. In the closed modality, our system achieves best performance in lemmatization and in classical subtask of POS tagging (where the texts from the training data authors are annotated), and second place in cross-genre and cross-time settings.

5. Ablation Experiments

The effect of various kinds contextualized embeddings is evaluated in Table 4. While BERT embeddings yield only a minor accuracy increase, which is consistent with (Straka et al., 2019b) for Latin, using XLM-RoBERTa leads to larger accuracy improvement. For comparison, we include the post-competition system with XLM-RoBERTa embeddings in Tables 2 and 3.

To quantify the boost of the additional training data in the open modality, we considered all models from the above mentioned Table 4, arriving at the average improvement presented in Table 5. While the performance on the in-domain test set (classical subtask) improves only slightly, the out-of-domain test sets (cross-genre and cross-time subtasks) show more substantial improvement with the additional training data.

The effect of different granularity of treebank embeddings in open modality is investigated in Table 6. When treebank embeddings are removed from our competition system, the performance deteriorates the most, even if only a little in absolute terms. This indicates that the UD and EvaLatin annotations are very consistent. Providing one embedding for EvaLatin data and another for all UD treebanks improves the performance, and more so if three UD treebank specific

Word embeddings	BERT embeddings	XLM-RoBERTa embeddings	Lemmatization			Tagging		
			classical	cross-genre	cross-time	classical	cross-genre	cross-time
Open modality								
✗	✗	✗	96.04	86.85	90.58	96.46	90.44	87.66
✓	✗	✗	96.27	87.28	90.80	96.64	91.16	87.78
✗	✓	✗	96.19	86.76	90.78	96.70	90.34	87.50
✗	✗	✓	96.33	86.48	90.95	96.80	90.67	87.79
✓	✓	✗	96.28	87.28	90.80	96.74	91.11	87.69
✓	✗	✓	96.35	87.48	91.07	96.82	91.46	87.91
Closed modality								
✗	✗	✗	95.62	84.62	87.63	96.14	88.90	83.59
✓	✗	✗	95.79	85.55	88.37	96.44	90.59	84.14
✗	✓	✗	95.65	84.76	87.58	96.44	89.08	84.84
✗	✗	✓	95.93	84.97	87.63	96.67	89.36	84.24
✓	✓	✗	95.96	85.52	88.04	96.65	90.15	84.93
✓	✗	✓	95.93	85.94	87.88	96.76	90.50	84.70

Table 4: The evaluation of various pretrained embeddings (FastText word embeddings, Multilingual BERT embeddings, XLM-RoBERTa embeddings) on the lemmatization and POS tagging.

	Lemmatization			Tagging		
	classical	cross-genre	cross-time	classical	cross-genre	cross-time
The improvement of open modality, i.e., using all three UD Latin treebanks	+0.430	+1.795	+2.975	+0.177	+1.100	+3.315

Table 5: The average percentage point improvement in the open modality settings compared to the closed modality. The results are averaged over all models in Table 4.

	Lemmatization			Tagging		
	classical	cross-genre	cross-time	classical	cross-genre	cross-time
Per-author embeddings, per-UD-treebank embeddings	96.28	87.28	90.80	96.74	91.11	87.69
Single EvaLatin embedding, per-UD-treebank embeddings	96.28	87.28	90.80	96.70	91.11	87.69
Single EvaLatin embedding, single UD-treebank embedding	96.23	87.22	90.78	96.68	91.14	87.63
EvaLatin and UD treebanks merged	96.18	87.23	90.77	96.52	91.01	86.12

Table 6: The effect of various kinds of treebank embeddings in open modality – whether the individual authors in EvaLatin get a different or the same treebank embedding, and whether the UD treebanks get a different treebank embedding, same treebank embedding but different from the EvaLatin data, or the same treebank embedding as EvaLatin data.

	Lemmatization	Tagging
	classical	classical
The improvement of using per-author treebank embeddings	0.027	0.043

Table 7: The average percentage point improvement of using per-author treebank embedding compared to not distinguishing among authors of EvaLatin data, averaged over all models in Table 4.

embeddings are used.

Lastly, we evaluate the effect of the per-author embeddings. While on the development set the improvement was larger, the results on the test sets are nearly identical. To get more accurate estimate, we computed the average improvement for all models in Table 4, arriving at marginal improvements in Table 7, which indicates that per-author embeddings have nearly no effect on the final system performance (compared to EvaLatin and UD specific embeddings).

6. Conclusion

We described our entry to the EvaLatin 2020 shared task, which placed first in the open modality and delivered strong performance in the closed modality.

For a future shared task, we think it might be interesting to include also segmentation and tokenization or extend the shared task with an extrinsic evaluation.

7. Acknowledgements

This work was supported by the grant no. GX20-16819X of the Grant Agency of the Czech Republic, and has been using language resources stored and distributed by the LINDAT/CLARIAH-CZ project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2018101).

8. Bibliographical References

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR*.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2019). Unsupervised Cross-lingual Representation Learning at Scale. *CoRR*, abs/1911.02116.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, pages 5–6.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November.

Lewis, P., Oguz, B., Rinott, R., Riedel, S., and Schwenk, H. (2019). Mlqa: Evaluating cross-lingual extractive question answering. *ArXiv*, abs/1910.07475.

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. (2015). Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. *CoRR*.

McCarthy, A. D., Vylomova, E., Wu, S., Malaviya, C., Wolf-Sonkin, L., Nicolai, G., Kirov, C., Silfverberg, M., Mielke, S. J., Heinz, J., Cotterell, R., and Hulden, M. (2019). The SIGMORPHON 2019 Shared Task: Morphological Analysis in Context and Cross-Lingual Transfer for Inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244, Florence, Italy, August. Association for Computational Linguistics.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Sprugnoli, R., Passarotti, M., Cecchini, F. M., and Pellegrini, M. (2020). Overview of the evalatin 2020 evaluation campaign. In Rachele Sprugnoli et al., editors, *Proceedings of the LT4HALA 2020 Workshop - 1st Workshop on Language Technologies for Historical and Ancient Languages, satellite event to the Twelfth International Conference on Language Resources and Evaluation (LREC 2020)*, Paris, France, May. European Language Resources Association (ELRA).

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 the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia. European Language Resources Association.

Straka, M., Straková, J., and Hajic, J. (2019a). UDPipe at SIGMORPHON 2019: Contextualized Embeddings, Regularization with Morphological Categories, Corpora Merging. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 95–103, Florence, Italy, August. Association for Computational Linguistics.

Straka, M., Straková, J., and Hajič, J. (2019b). Evaluating Contextualized Embeddings on 54 Languages in POS Tagging, Lemmatization and Dependency Parsing. *arXiv e-prints*, page arXiv:1908.07448, August.

Straka, M. (2018). UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task. In *Proceedings of CoNLL 2018: The SIGNLL Conference on Computational Natural Language Learning*, pages 197–207, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stymne, S., de Lhoneux, M., Smith, A., and Nivre, J. (2018). Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625, Melbourne, Australia, July. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Zeman, D., Popel, M., Straka, M., Hajič, J., Nivre, J., Ginter, F., et al. (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, Vancouver, Canada. Association for Computational Linguistics.

Zeman, D., Hajič, 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 1–20, Brussels, Belgium, October. Association for Computational Linguistics.

9. Language Resource References

Ginter, F., Hajič, J., Luotolahti, J., Straka, M., and Zeman, D. (2017). *CoNLL 2017 Shared Task - Automatically*

Annotated Raw Texts and Word Embeddings. Institute of Formal and Applied Linguistics, LINDAT/CLARIN, Charles University, Prague, Czech Republic, LINDAT/CLARIN PID: http://hdl.handle.net/11234/1-1989.

Zeman, D., Nivre, J., et al. (2019). *Universal Dependencies 2.5*. Institute of Formal and Applied Linguistics, LINDAT/CLARIN, Charles University, Prague, Czech Republic, LINDAT/CLARIN PID: http://hdl.handle.net/11234/1-3105.