TurkuNLP: Delexicalized Pre-training of Word Embeddings for Dependency Parsing

Jenna Kanerva[1,2], **Juhani Luotolahti**[1,2], **and Filip Ginter**[1]
[1]Turku NLP Group
[2]University of Turku Graduate School (UTUGS)
University of Turku, Finland
jmnybl@utu.fi, mjluot@utu.fi, figint@utu.fi

Abstract

We present the TurkuNLP entry in the *CoNLL 2017 Shared Task on Multilingual Parsing from Raw Text to Universal Dependencies*. The system is based on the UDPipe parser with our focus being in exploring various techniques to pre-train the word embeddings used by the parser in order to improve its performance especially on languages with small training sets. The system ranked 11th among the 33 participants overall, being 8th on the small treebanks, 10th on the large treebanks, 12th on the parallel test sets, and 26th on the surprise languages.

1 Introduction

In this paper we describe the TurkuNLP entry in the *CoNLL 2017 Shared Task on Multilingual Parsing from Raw Text to Universal Dependencies* (Zeman et al., 2017). The Universal Dependencies (UD) treebank collection (Nivre et al., 2017b) has 70 treebanks for 50 languages with cross-linguistically consistent annotation. Of these, the 63 treebanks which have at least 10,000 tokens in their test section are used for training and testing the systems. Further, a parallel corpus consisting of 1,000 sentences in 14 languages was developed as an additional test set, and finally, the shared task included test sets for four "surprise" languages not known until a week prior to the test phase of the shared task (Nivre et al., 2017a). No training data was provided for these languages — only a handful of sentences was given as an example. As an additional novelty, participation in the shared task involved developing an end-to-end parsing system, from raw text to dependency trees, for all of the languages and treebanks. The participants were provided with automatically predicted word and sentence segmentation as well as morphological tags for the test sets, which they could choose to use as an alternative to developing own segmentation and tagging. These baseline segmentations and morphological analyses were provided by UDPipe v1.1 (Straka et al., 2016).

In addition to the manually annotated treebanks, the shared task organizers also distributed a large collection of web-crawled text for all but one of the languages in the shared task, totaling over 90 billion tokens of fully dependency parsed data. Once again, these analyses were produced by the UDPipe system. This automatically processed large dataset was intended by the organizers to complement the manually annotated data and, for instance, support the induction of word embeddings.

As an overall strategy for the shared task, we chose to build on an existing parser and focus on exploring various methods of pre-training the parser and especially its embeddings, using the large, automatically analyzed corpus provided by the organizers. We expected this strategy to be particularly helpful for languages with only a little training data. On the other hand we put only a minimal effort into the surprise languages. We also chose to use the word and sentence segmentation of the test datasets, as provided by the organizers. As we will demonstrate, the results of our system correlate with the focus of our efforts. Initially, we focused on the latest ParseySaurus parser (Alberti et al., 2017), but due to the magnitude of the task and restrictions on time, we finally used the UDPipe parsing pipeline of Straka et al. (2016) as the basis of our efforts.

2 Word Embeddings

The most important component of our system is the novel techniques we used to pre-train the

word embeddings. Our word embeddings combine three important aspects: 1) delexicalized syntactic contexts for inducing word embeddings, 2) word embeddings built from character n-grams, and 3) post-training injection and modification of embeddings for unseen words.

2.1 Delexicalized Syntactic Contexts

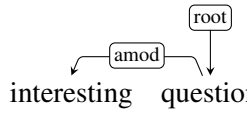
Word embeddings induced from large text corpora have been a key resource in many NLP task in recent years. In many common tools for learning word embeddings, such as word2vec (Mikolov et al., 2013), the context for a focus word is a sliding window of words surrounding the focus word in linear order. Levy and Goldberg (2014) extend the context with dependency trees, where the context is defined as the words nearby in the dependency tree with additionally the dependency relation attached to the context words, for example *interresting/amod*.

In Kanerva et al. (2017), we show that word embeddings trained in a strongly syntactic fashion outperform standard word2vec embeddings in dependency parsing. In particular, the context is fully delexicalized — instead of using words in the word2vec output layer, only part-of-speech tags, morphological features and syntactic functions are predicted. This delexicalized syntactic context is shown to lead to higher performance as well as generalize better across languages.

In our shared task submission we build on top of the previous work and optimize the embeddings even closer to the parsing task: We extend the original delexicalized context to also predict parsing actions of a transition-based parser. From the existing parse trees in the raw data collection, we create the transition sequence used to produce the tree and for each word collect features describing the actions taken when the word is on top-2 positions of the stack, e.g. if the focus word is first on stack, what is the next action. Our word–context pairs are illustrated in Table 1. In this way, we strive to build embeddings which relate together words which appear in similar configurations of the parser.

2.2 Word embeddings from character n-grams

It is known that the initial embeddings affect the performance of neural dependency parsers and pre-training the embedding matrix prior to training has an important effect on the performance of neural systems (Chen and Manning, 2014; Andor et al., 2016).



input	context
interesting	ADJ
interesting	Degree=Pos
interesting	amod
question	NOUN
question	Number=Sing
question	root
question	Dep_amod
interesting	stack1_shift
interesting	stack2_left-arc
interesting	stack2_left_amod
question	stack1_left-arc
question	stack1_left_amod
question	stack1_root
character four-grams	
interesting, char_$int, char_inte, char_nter, char_tere, char_eres, char_rest, char_esti, char_stin, char_ting, char_ing$	
question, char_$que, char_ques, char_uest char_esti, char_stio, char_tion, char_ion$	

Table 1: Delexicalized contexts and character n-grams for word embeddings

Since the scope of languages in the task is large, the embeddings used for this dependency parsing task need to be able to be representative of languages with both large and small available resources, and also they need to be able to capture morphological information for languages with complex morphologies as well as those with less morphological variation.

To address better the needs of small languages with very little of data available as well as morphologically rich languages, we build our word embedding models with methods used in the popular fastText[1] representation of Bojanowski et al. (2016). They suggest, instead of tokens, to use character n-grams as the basic units in building embeddings for words. First, words are turned into character n-grams and embeddings are learned separately for each of these character n-grams. Secondly, word vectors are assembled from these

[1] https://github.com/facebookresearch/fastText

trained character embeddings by averaging all character n-grams present in a word. To make the embeddings more informative, the n-grams include special markers for the beginning and the end of the token, allowing the model for example to learn special embeddings for word suffixes which are often used as inflectional markers. In addition to the n-grams, a vector for the full word is also trained, and when final word vectors are produced, the full word vector is treated similarly as other n-grams and is averaged as part of the final product along with the rest. This allows for the potential benefits of token level embeddings to be materialized in our model.

Table 1 demonstrates the splitting of a word into character four-grams with special start and end markers. When learning embeddings for these character n-grams, context for each n-gram is replicated from the original word context, i.e. each character n-gram created from the word *interesting* gets the delexicalized context assigned for that word, namely *ADJ, Degree=Pos, amod, stack1_shift, stack2_left-arc, stack2_left_amod* in the example Table 1.

This procedure offers multiple advantages. One of them is the ability to construct embeddings for previously unseen words, a common occurrence especially with languages with small training corpora. With these character n-gram embeddings we are basically able to build an embedding for any word, except very few cases where we do not find any of the character n-grams from our trained model. Another advantage of this embedding scheme is its better ability to address the morphological variation compared to plain token based embeddings.

2.3 Data and Parameters

For the training of word embeddings for each language we took training data from the treebank training section and the automatically analyzed raw corpus (Ginter et al., 2017). Using also the treebank training section is important for very small languages where there is very little of raw data, especially for Old Church Slavonic where the raw data has only 29,000 tokens compared to 37,500 tokens in the treebank training set, while for big languages it barely makes any difference as the treebank data gets buried under the mass of raw data. For each language we build character n-gram embedding models using word2vecf software[2] by Levy and Goldberg (2014) with negative sampling, skip-gram architecture, embeddings dimensionality of 100, delexicalized syntactic contexts explained in Section 2.1 and character n-grams of length 3-6. The maximum size of raw data used is limited to 50 million unique sentences in order to keep the training times bearable, and sentences longer that 30 tokens are discarded. Languages with only very limited resources we run 10 training iterations, but for rest only one iteration is used. These character n-gram models explained in detail in Section 2.2 can then be used to build word embeddings for arbitrary words, only requiring that at least one of the extracted character n-grams is present in our embedding model.

For parsing we also used the parameters optimized for UDPipe baseline system and changed only parts related to pre-trained embeddings. As we do not perform further parameter optimization, we trained our models always using the full training set, also in cases where different development sets were not provided. For small treebanks without development data, we did not test our models in advance but trusted methods tested on other treebanks to generalize also for these. For each language we include 100-dim word embeddings trained using our methods described in Sections 2.1 and 2.2. Additionally, pre-trained feature embeddings are trained for upos+feat combinations included in the xpostag column. These embeddings are trained using the transition actions as delixicalized context, and vectors for full feature combinations are constructed from individual features using the same character n-gram method as in the word embeddings (one feature is now the same as one character n-gram).

3 Parsing Pipeline

Our submission builds on top of the UDPipe parsing pipeline by Straka et al. (2016). We use data segmented by the UDPipe baseline systems as our system input, and then morphological analysis and syntactic parses are produced with our own UDPipe models. The UDPipe morphological tagger (MorphoDiTa (Straková et al., 2014)) is run as-is with parameters optimized for the baseline system (released together with the baseline models). The only exception is that we replaced the language-specific postag (*xpostag*) column with a combined

[2] https://github.com/BIU-NLP/word2vecf

universal postag (*upostag*) and morphological features (*feats*), and trained the tagger to produce this instead of the language-specific postags. We did not expect this to affect the tagging accuracy, but instead it was used to provide pre-trained feature embeddings for the parser.

We further modified the UDPipe parser to allow including new embedding matrices after the model has been trained. This gives us an easy way to add embeddings for arbitrary words without a need of training a new parsing model. As we are able to create word embeddings for almost any word using our character n-gram embeddings described in Section 2.2, we are able to collect vocabulary from the data we are parsing, create vectors for previously unseen words and inject these vectors into the parsing model. This method essentially eliminates all out of vocabulary words.

3.1 Post-training modifications of word embeddings

The UDPipe parser uses the common technique of adjusting the word embeddings during training. The magnitude of the change imposed by the parser depends on the frequency of the word in the training data and, naturally, only words seen in the training set are subject to this training-phase adjustment. Therefore, we implemented a step whereby we transfer the parser adjustments onto the words not seen in the training data. For every such unseen word, we calculate its translation in the vector space by summing the parser-induced changes of known words in its neighborhood. These are weighted by their cosine similarity with the unknown word, using a linear function mapping similarities in the $[0.5, 1.0]$ interval into weights in the $[0.0, 1.0]$ range. I.e. known words with similarity below 0.5 do not contribute, and thereafter the weighting is linear. The overall effect of this modification observed in our development runs was marginal.

3.2 Parallel test sets (PUD)

Parallel test sets are parsed with a model trained on the default treebank of a language (the one without any treebank-specific suffix). For many languages only one treebank exists and no choice is needed, but for some there are two or even more choices. We chose to use these treebanks without treebank suffixes as the very first treebank included for a language will receive just the language code without the treebank suffix while newer treebanks will get a distinguishable treebank suffix. It then means that the default treebanks without suffixes have been part of the UD collection longer, many of these originating from the Google's Universal Treebank collection (McDonald et al., 2013). We hypothesized these treebanks to be more harmonized to the UD guidelines and apply better to the new test sets.

3.3 Surprise languages

In this work we did not concentrate on parsing the four surprise languages, and only used a very naive approach to complete the submission of all required test sets. For each surprise language we simply picked one existing model among all models trained for regular treebanks. We parsed the small sample of example sentences (about 20 sentences for each language) with all existing models, and picked the one which maximized the LAS score (Kazakh for Buryat, Galician-TreeGal for Kurmanji, Portuguese for North Sami and Slovenian for Upper Sorbian) without doing any treebank size, language family or related language evaluation. The only change in the parsing model is that during parsing, we mask all word embeddings, this way preventing the parser to use the vector for unknown word too often. This makes our parsing model delexicalized as all word embeddings are zeroed after training and not used in parsing, with the exception that parsing model is trained using information from word embeddings.

4 Results

The participating systems are tested using TIRA platform (Potthast et al., 2014), where the system must be deployed on a virtual machine and the test sets are processed without direct access to the data. Overall rank of our system in the official evaluation is 11th out of 33 participating teams with a macro-averaged labeled attachment score (LAS) of 68.59%.[3] On macro-LAS score across all treebanks, we are clearly behind the winning system (Stanford, 76.30%), but our pre-trained word embeddings are able to improve over the baseline UDPipe by 0.24% points on average. When looking only at treebanks with very little of training data we gain on average 2.38% over the baseline system. The same number for the big treebanks

[3] Full list of the official results can be found at http://universaldependencies.org/conll17/results.html.

only is +1.15%, +0.23% for parallel test sets and -16.55% for surprise languages. Based on these numbers we can clearly see that we managed to get a substantial improvement over the baseline system on very small languages where we also assumed our word embeddings to be most helpful. Instead, our very naive approach for handling surprise languages is clearly not sufficient, and a better approach should have been implemented. Detailed results of our system are shown in Table 2.

On official evaluation our system ranked sixth on universal part-of-speech tagging and second on morphological features. We see modest improvement of +0.22% (*upos*) and +0.27% (*feats*) over the baseline models. As word embeddings are not used in tagging and we use the same parameters as the baseline system, the only modification we did in tagging is that instead of using language-specific postags (*xpostag*), we concatenated universal postag and morphological features into xpostag column and trained the tagger to produce this concatenation.

5 Conclusions

We have presented our entry in the *CoNLL 2017 Shared Task on Multilingual Parsing from Raw Text to Universal Dependencies*, with the 11th rank of 33 participants. During the development, we have focused on exploring various ways of pre-training the embeddings used by the parser, as well as providing embeddings also for the unknown words in the data to be parsed. In particular, we have proposed a method of pre-training the embeddings using an entirely delexicalized output layer of the word2vec skip-gram model. This mode of pre-training the embeddings is shown to be superior to the usual approach of pre-training with the standard word2vec skip-gram with negative sampling. We have also explored, here with only a minimal improvement to the score, the possibility of post-hoc application of the adjustments effected by the parser on the word embeddings during the training phase. All our components used in this paper are freely available at `https://github.com/TurkuNLP/conll17-system`.

Acknowledgments

This work was supported by the Finnish Academy and the Kone Foundation. Computational resources were provided by CSC – IT Center for Science, Finland.

References

Chris Alberti, Daniel Andor, Ivan Bogatyy, Michael Collins, Dan Gillick, Lingpeng Kong, Terry Koo, Ji Ma, Mark Omernick, Slav Petrov, et al. 2017. Syntaxnet models for the conll 2017 shared task. *arXiv preprint arXiv:1703.04929* .

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042* .

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*. pages 740–750.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-1989.

Jenna Kanerva, Sampo Pyysalo, and Filip Ginter. 2017. Delexicalized contexts for pure dependency-based word embeddings. In *Proceedings of the International Conference on Dependency Linguistics (Depling'17)*. Under review.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*. pages 92–97.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017a. Universal dependencies 2.0 CoNLL 2017 shared task development and test data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-2184.

Joakim Nivre et al. 2017b. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague, `http://hdl.handle.net/11234/1-1983`. http://hdl.handle.net/11234/1-1983.

Big Treebanks

Treebank	Rank	LAS F1	Treebank	Rank	LAS F1
ar	17	65.74	hr	9	78.57
bg	11	84.85	hu	10	65.61
ca	11	85.64	id	12	74.87
cs	13	83.48	it	15	85.66
cs_cac	8	84.28	ja	15	72.81
cs_cltt	11	73.83	ko	11	66.93
cu	11	65.43	la_ittb	9	78.99
da	12	74.61	la_proiel	11	59.86
de	17	69.32	lv	8	62.13
el	13	79.93	nl	14	69.59
en	11	76.68	nl_lassysmall	14	79.06
en_lines	8	74.77	no_bokmaal	13	83.60
en_partut	13	74.48	no_nynorsk	10	82.35
es	15	81.79	pl	12	80.11
es_ancora	15	84.15	pt	10	82.91
et	10	59.79	pt_br	12	86.36
eu	13	70.22	ro	10	80.71
fa	20	76.54	ru	17	74.69
fi	9	75.82	ru_syntagrus	13	86.79
fi_ftb	9	75.59	sk	12	74.72
fr	12	80.61	sl	8	82.77
fr_sequoia	10	81.12	sv	12	77.35
gl	17	77.66	sv_lines	13	74.46
got	9	61.52	tr	13	54.69
grc	8	59.83	ur	14	77.06
grc_proiel	7	68.04	vi	14	38.07
he	16	57.50	zh	12	58.71
hi	7	87.75			
			All	10	74.19

Parellel Test Sets

Treebank	Rank	LAS F1
ar_pud	27	42.34
cs_pud	11	80.02
de_pud	17	66.78
en_pud	9	79.61
es_pud	12	78.02
fi_pud	8	79.61
fr_pud	14	74.17
hi_pud	8	51.87
it_pud	12	84.18
ja_pud	19	76.09
pt_pud	12	74.09
ru_pud	8	69.11
sv_pud	16	69.90
tr_pud	15	34.09
All	12	68.56

All treebanks

Treebank	Rank	LAS F1
All	11	68.59

Surprise Test Sets

Treebank	Rank	LAS F1
bxr	26	14.22
hsb	25	34.67
kmr	23	22.19
sme	27	10.99
All	26	20.52

Small Treebanks

Treebank	Rank	LAS F1
fr_partut	12	78.83
ga	7	64.25
gl_treegal	9	66.47
kk	2	28.31
la	9	47.91
la_ittb	9	78.99
la_proiel	11	59.86
sl_sst	11	47.50
ug	6	36.51
uk	7	63.70
All	8	54.18

Table 2: Rank and labeled attachment score for our system in the official evaluation.

Martin Potthast, Tim Gollub, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2014. Improving the reproducibility of PAN's shared tasks: Plagiarism detection, author identification, and author profiling. In Evangelos Kanoulas, Mihai Lupu, Paul Clough, Mark Sanderson, Mark Hall, Allan Hanbury, and Elaine Toms, editors, *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14)*. Springer, Berlin Heidelberg New York, pages 268–299. https://doi.org/10.1007/978-3-319-11382-1_22.

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

Jana Straková, Milan Straka, and Jan Hajič. 2014. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 13–18. http://www.aclweb.org/anthology/P/P14/P14-5003.pdf.

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, Hěctor Martínez Alonso, 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.