Turku Neural Parser Pipeline: An End-to-End System for the CoNLL 2018 Shared Task

Jenna Kanerva[1,2] **Filip Ginter**[1] **Niko Miekka**[1] **Akseli Leino**[1] **Tapio Salakoski**[1]
[1]Turku NLP Group, Department of Future Technologies, University of Turku, Finland
[2]University of Turku Graduate School (UTUGS)
firstname.lastname@utu.fi

Abstract

In this paper we describe the TurkuNLP entry at the *CoNLL 2018 Shared Task on Multilingual Parsing from Raw Text to Universal Dependencies*. Compared to the last year, this year the shared task includes two new main metrics to measure the morphological tagging and lemmatization accuracies in addition to syntactic trees. Basing our motivation into these new metrics, we developed an end-to-end parsing pipeline especially focusing on developing a novel and state-of-the-art component for lemmatization. Our system reached the highest aggregate ranking on three main metrics out of 26 teams by achieving 1st place on metric involving lemmatization, and 2nd on both morphological tagging and parsing.

1 Introduction

The 2017 and 2018 CoNLL UD Shared tasks aim at an evaluation of end-to-end parsing systems on a large set of treebanks and languages. The 2017 task (Zeman et al., 2017) focused primarily on the evaluation of the syntactic trees produced by the participating systems, whereas the 2018 task (Zeman et al., 2018) adds further two metrics which also measure the accuracy of morphological tagging and lemmatization. In this paper, we present the TurkuNLP system submission to the *CoNLL 2018 UD Shared Task*. The system is an end-to-end parsing pipeline, with components for segmentation, morphological tagging, parsing, and lemmatization. The tagger and parser are based on the 2017 winning system by Dozat et al. (2017), while the lemmatizer is a novel approach utilizing the OpenNMT neural machine translation system for sequence-to-sequence learning. Our pipeline ranked first on the evaluation metric related to lemmatization, and second on the metrics related to tagging and parsing.

2 Task overview

CoNLL 2018 UD Shared Task is a follow-up to the 2017 shared task of developing systems predicting syntactic dependencies on raw texts across a number of typologically different languages. In addition to the 82 UD treebanks for 57 languages, which formed the primary training data, the participating teams were allowed to use also additional resources such as Wikipedia dumps[1], raw web crawl data and word embeddings (Ginter et al., 2017), morphological transducers provided by Apertium[2] and Giellatekno[3], and the OPUS parallel corpus collection (Tiedemann, 2012). In addition to the 2017 primary metric (LAS), the systems were additionally evaluated also on metrics which include lemmatization and morphology prediction. In brief, the three primary metrics of the task are as follows (see Zeman et al. (2018) for detailed definitions):

LAS The proportion of words which have the correct head word with the correct dependency relation.

MLAS Similar to LAS, with the additional requirement that a subset of the morphology features is correctly predicted and the functional dependents of the word are correctly attached. MLAS is only calculated on content-bearing words, and strives to level the field w.r.t. morphological richness of languages.

[1]https://dumps.wikimedia.org
[2]https://svn.code.sf.net/p/apertium/svn/languages
[3]https://victorio.uit.no/langtech/trunk/langs

Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 133–142
Brussels, Belgium, October 31 – November 1, 2018.
https://doi.org/10.18653/v1/K18-2013

BLEX The proportion of head-dependent content word pairs whose dependency relation and both lemmas are correct.

3 System overview and rationale

The design of the pipeline was dictated by the tight schedule and the limited manpower we were able to invest into its development. Our overall objective was to develop an easy-to-use parsing pipeline which carries out all the four tasks of segmentation, morphological tagging, parsing, and lemmatization, resulting in an end-to-end full parsing pipeline reusable in downstream applications. We also strove for the pipeline to perform well on all four tasks and all groups of treebanks, ranging from the large treebanks to the highly under-resourced ones. With this in mind, we decided to rely on openly available components when the acceptable performance is already met, and create our own components for those tasks we see clear room for improvement.

Therefore, for segmentation, tagging and parsing we leaned as much as possible on well-known components trained in the standard manner, and deviated from these only when necessary. Our approach to lemmatization, on the other hand, is original and previously unpublished. In summary, we rely for most but not all languages on the tokenization and sentence splitting provided by the UDPipe baseline (Straka et al., 2016). Tagging and parsing is carried out using the parser of Dozat et al. (2017), the winning entry of the 2017 shared task. Using a simple data manipulation technique, we also obtain the morphological feature predictions from the same tagger which was originally used to produce only universal part-of-speech (`UPOS`) and language-specific part-of-speech (`XPOS`) predictions. Finally, the lemmatization is carried out using the OpenNMT neural machine translation toolkit (Klein et al., 2017), casting lemmatization as a machine translation problem. All these components are wrapped into one parsing pipeline, making it possible to run all four steps with one simple command and gain state-of-the-art or very close to state-of-the-art results for each step. In the following, we describe each of these four steps in more detail, while more detailed description of the pipeline itself is given in Section 6.

3.1 Tokenization and sentence splitting

For all but three languages, we rely on the UDPipe baseline runs provided by the shared task organizers. The three languages where we decided to deviate from the baseline are Thai, Breton and Faroese. Especially for Thai we suspected the UDPipe baseline, trained without ever seeing a single character of the Thai alphabet, would perform poorly. For Breton, we were unsure about the way in which the baseline system tokenizes words with apostrophes like *arc'hant* (money), and without deeper knowledge of Breton language decided that it is better to explicitly keep all words with apostrophes unsegmented. We therefore developed a regular-expression based sentence splitter and tokenizer — admittedly under a very rushed schedule — which splits sentences and tokens on a handful of punctuation characters. While, after the fact, we can see that the UDPipe baseline performed well at 92.3%, our solution outperformed it by two percentage points, validating our choice. For Thai, we developed our own training corpus using machine translation (described later in the paper in Section 4.3), and trained UDPipe on this corpus, gaining a segmentation model at the same time. Indeed, the UDPipe baseline only reached 8.5% accuracy while our tokenizer performed at the much higher 43.2% (still far below the 70% achieved by the Uppsala team). Similarly, for Faroese we built training data by pooling the Danish-DDT, Swedish-Talbanken, and the three available Norwegian treebanks (Bokmaal, Nynorsk, NynorskLIA), and subsequntly trained the UDPipe tokenizer on this data. After the fact, we can see that essentially all systems performed in the 99–100% range on Faroese, and we could have relied on the UDPipe baseline.

On a side note, we did develop our own method for tokenization and sentence splitting but in the end, unsure about its stability and performance on small treebanks, we decided to "play it safe" and not include it in the final system. However, the newly developed tokenizer is part of our open-source pipeline release and trainable on new data.

3.2 Pre-trained embeddings

Where available, we used the pre-trained embeddings from the 2017 shared task (Ginter et al., 2017). Embeddings for Afrikaans, Breton, Buryat, Faroese, Gothic, Upper Sorbian, Armenian, Kurdish, Northern Sami, Serbian and Thai were ob-

tained from the embeddings published by Facebook[4] trained using the fastText method (Bojanowski et al., 2016), and finally for Old French (Old French-SRCMF) we took the embeddings trained using word2vec (Mikolov et al., 2013) on the treebank train section by the organizers in their baseline UDPipe model release. We did not pretrain any embeddings ourselves.

3.3 UPOS tagging

UPOS tagging for all languages is carried out using the system of Dozat et al. (2017) trained out-of-the-box with the default set of parameters from the CoNLL-17 shared task. The part-of-speech tagger is a time-distributed affine classifier over tokens in a sentence, where tokens are first embedded with a word encoder which sums together a learned token embedding, a pre-trained token embedding and a token embedding encoded from the sequence of its characters using unidirectional LSTM. After that bidirectional LSTM reads the sequence of embedded tokens in a sentence to create a context-aware token representations. These token representations are then transformed with ReLU layers separately for each affine tag classification layers (namely UPOS and XPOS). These two classification layers are trained jointly by summing their cross-entropy losses. For more detailed description, see Dozat and Manning (2016) and Dozat et al. (2017).

3.4 XPOS and FEATS tagging

As the tagger of Dozat et al. predicts the XPOS field, we used a simple trick of concatenating the `FEATS` field into `XPOS`, therefore manipulating the tagger into predicting the XPOS and morphological features as one long string. For example the original `XPOS` field value `N` and `FEATS` field value `Case=Nom|Number=Sing` in Finnish-TDT treebank gets concatenated into `XPOS=N|Case=Nom|Number=Sing` and this full string is predicted as one class by the tagger. After tagging and parsing, these values are again splitted into correct columns. This is a (embarrassingly) simple approach which leads to surprisingly good results, as our system ranks 3rd in morphological features with accuracy of 86.7% over all treebanks, 0.9pp below the Uppsala team which ranked 1st on this subtask.

[4] `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`

We, in fact, did at first develop a comparatively complex morphological feature prediction component which outperformed the state-of-the-art on the 2017 shared task, but later we discovered that the simple technique described above somewhat surprisingly gives notably better results. We expected that the complex morphology of many languages leads to a large number of very rare morphological feature strings, a setting unsuitable for casting the problem as a single multi-class prediction task. Consequently, our original attempt at morphological tagging predicted value for each morphological category separately from a shared representation layer, rather than predicting the full feature string at once. To shed some light on the complexity of the problem in terms of the number of classes, and understand why a multi-class setting works well, we list in Table 1 the number of unique morphological feature strings needed to cover 80%, 90%, 95%, and 100% of the running words in the training data for each language. The number of unique feature combinations varies from 15 (Japanese-GSD, Vietnamese-VTB) to 2629 (Czech-PDT), and for languages with high number of unique combinations, we can clearly see that there is a large leap from covering 95% of running words to covering full 100%. For example in Czech-PDT, only 349 out of the 2629 feature combinations are needed to cover 95% of running words, and the rest 2280 (of which 588 are singletons) together accounts only 5% of running words. Based on these numbers our conclusions are that a focus on predicting the rare feature combinations correctly does not affect the accuracy much, and learning a reasonable number of common feature combinations well seems to be a good strategy in the end.

Interestingly, on our preliminary experiments with Finnish, we found that concatenating `FEATS` into `XPOS` improved also LAS by more than 0.5pp, since the parser takes the `XPOS` field as a feature and benefits from the additional morphological information present. To investigate this more closely and test whether the same improvement can be seen on other languages as well, we carry out an experiment where we train the tagger and parser without morphological information for Finnish and six more arbitrarily chosen treebanks. This new experiment then follows the original training setting used by the Stanford team on their CoNLL-17 submission, and by comparing this to

	80%	*90%*	*95%*	*100%*
Czech-PDT	96	194	349	2629
Finnish-TDT	79	188	349	2052
Finnish-FTB	72	174	333	1762
Czech-CAC	81	160	285	1745
Czech-FicTree	73	161	287	1464
Slovak-SNK	79	163	283	1199
Ukrainian-IU	91	186	322	1197
Polish-LFG	84	170	281	1171
Slovenian-SSJ	73	141	254	1101
Croatian-SET	63	125	212	1099
Latin-PROIEL	121	214	323	1031
Ancient Greek-PROIEL	114	203	308	1027
Urdu-UDTB	30	61	124	1001
Polish-SZ	80	157	267	991
Latin-ITTB	58	136	226	985
Turkish-IMST	54	139	262	972
Hindi-HDTB	38	76	127	939
Estonian-EDT	43	89	151	918
German-GSD	58	96	141	909
Basque-BDT	51	100	169	884
Old Church Slavonic-PROIEL	78	168	276	859
Latvian-LVTB	57	119	218	828
Ancient Greek-Perseus	59	107	169	774
Russian-SynTagRus	67	124	176	734
Slovenian-SST	73	146	233	645
Gothic-PROIEL	75	138	214	623
Hungarian-Szeged	40	90	166	581
Serbian-SET	48	85	131	539
Hebrew-HTB	19	45	85	521
Romanian-RRT	34	58	97	451
Bulgarian-BTB	33	63	107	432
Latin-Perseus	58	100	144	418
Portuguese-Bosque	20	35	60	396
Russian-Taiga	66	126	182	376
North Sami-Giella	39	78	127	369
Irish-IDT	47	81	125	360
Greek-GDT	57	90	123	348

	80%	*90%*	*95%*	*100%*
Arabic-PADT	22	35	53	322
Spanish-AnCora	28	48	71	295
Italian-ISDT	22	35	55	281
Catalan-AnCora	28	47	68	267
French-GSD	19	31	46	225
Italian-PoSTWITA	23	39	56	224
Galician-TreeGal	23	41	66	222
Uyghur-UDT	21	40	63	214
Swedish-Talbanken	26	43	61	203
Norwegian-Bokmaal	26	39	57	203
French-Sequoia	25	43	62	200
Indonesian-GSD	12	20	31	192
Norwegian-Nynorsk	26	41	53	184
Swedish-LinES	25	43	61	173
Persian-Seraji	11	19	31	162
Danish-DDT	24	38	53	157
Armenian-ArmTDP	51	85	117	157
English-EWT	19	32	45	150
Upper Sorbian-UFAL	48	88	111	134
English-LinES	18	29	43	104
English-GUM	16	27	40	104
Kazakh-KTB	29	49	71	98
Norwegian-NynorskLIA	22	34	46	96
Dutch-Alpino	16	24	31	63
Afrikaans-AfriBooms	14	22	28	61
Dutch-LassySmall	13	19	26	59
Kurmanji-MG	24	35	46	58
Old French-SRCMF	11	15	19	57
Buryat-BDT	17	26	34	41
Chinese-GSD	7	10	13	31
Galician-CTG	7	9	11	27
Korean-GSD	4	4	6	19
Korean-Kaist	6	8	10	17
French-Spoken	8	10	12	16
Vietnamese-VTB	6	8	10	15
Japanese-GSD	5	7	9	15

Table 1: The number of unique UPOS+morphlogical feature combinations needed to cover 80%, 90%, 95% and 100% of the running words in each treebank.

our main runs we can directly evaluate the effect of predicting additional morphological information. Three of the treebanks used in this experiment (Arabic-PADT, Czech-PDT and Swedish-Talbanken) seem to originally encode the full (or at least almost full) morphological information in the XPOS field in a language-specific manner (e.g. `AAFS1----2A----` in Czech), whereas four treebanks seem to include only part-of-speech like information or nothing at all in the XPOS field (Estonian-EDT, Finnish-TDT, Irish-IDT and Russian-SynTagRus).

The results of this experiment are shown in Table 2. Four treebanks above the dashed line, those originally including only part-of-speech like information in the `XPOS` field, shows clear positive improvement in terms of LAS when the parser is able to see also morphological tags predicted together with the language-specific `XPOS`. The parser seeing the morphological tags (LAS$_m$ column) shows improvements approx. from +0.3 to +0.9 for these four treebanks compared to the parser without morphological tags (LAS column). Three treebanks below the dashed line, those already including language-specific morphological information in the `XPOS` field, quite naturally does not benefit from additional morphology and shows mildly negative results in terms of LAS. However the difference in treebanks showing negative results is substantially smaller compared to those having positive effect (negative differences stay between -0.0 to -0.2), therefore based on these seven tree-

Treebank	LAS	LAS_m		UPOS	$UPOS_m$		XPOS	$XPOS_m$	
Estonian-EDT	83.40	**84.15**	(+0.75)	96.32	**96.45**	(+0.13)	97.81	**97.87**	(+0.06)
Finnish-TDT	85.74	**86.60**	(+0.86)	96.45	**96.66**	(+0.21)	97.48	**97.63**	(+0.15)
Irish-IDT	70.01	**70.88**	(+0.87)	91.87	**92.36**	(+0.49)	91.01	**91.05**	(+0.04)
Russian-SynT.	91.40	**91.72**	(+0.32)	**98.11**	98.03	(-0.08)	—	—	
Arabic-PADT	**72.67**	72.45	(-0.22)	90.39	**90.48**	(+0.19)	87.36	**87.39**	(+0.03)
Czech-PDT	**90.62**	90.57	(-0.05)	**98.76**	98.74	(-0.02)	**95.66**	95.44	(-0.22)
Swedish-Talb.	**85.87**	85.83	(-0.04)	97.40	**97.47**	(+0.07)	96.36	**96.41**	(+0.05)

Table 2: LAS, UPOS and XPOS scores for seven parsers trained with and without tagger predicting the additional morphological information. *m* after the score name stands for including the morphological information during training, i.e. the official result for our system. Note that when evaluating `XPOS`, the morphological information is already extracted from that field so the evaluation only includes prediction of original XPOS-tags, not morphological features.

banks the overall impact stays on positive side. Note that during parsing the parser only sees predicted morphological features, so this experiment confirms that predicting more complex information on lower-level can improve the parser.

Because of the fact that many treebanks include more than plain part-of-speech information in the language-specific `XPOS` field, likely more natural place for the morphological features would be the universal part-of-speech field `UPOS` which is guaranteed to include only universal part-of-speech information. However, with the limited time we had during the shared task period, we had no time to test whether adding morphological features harms the prediction of original part-of-speech tag, and we decided to use `XPOS` field as we thought it's least important of these two. Based on the results in the XPOS column of Table 2, we however see that additional information does not generally seem to harm the prediction of the original language-specific part-of-speech tags and hints towards the conclusion that likely the `UPOS` field could have been used with comparable performance.

3.5 Syntactic parsing

Syntactic parsing for all languages is carried out using the system of Dozat et al. trained out-of-the-box with the default set of parameters from the CoNLL-17 shared task. The parser architecture is quite similar as used in the tagger. Tokens are first embedded with a word encoder which sums together a learned token embedding, a pre-trained token embedding and a token embedding encoded from the sequence of its characters using unidirectional LSTM. These embedded tokens are yet concatenated together with corresponding part-of-speech embeddings. After that bidirectional LSTM reads the sequence of embedded tokens in a sentence to create a context-aware token representations. These token representations are then transformed with four different ReLU layers separately for two different biaffine classifiers to score possible relations (`HEAD`) and their dependency types (`DEPREL`), and best predictions are later decoded to form a tree. These relation and type classifiers are again trained jointly by summing their cross-entropy losses. For more detailed description, see Dozat and Manning (2016) and Dozat et al. (2017).

3.6 Lemmatization

While in many real word industry applications especially for inflective languages the lemmatizer is actually the most needed component of the parsing pipeline, yet it's performance has been undesirable weak in previous state-of-the-art parsing pipelines for many inflectionally complex languages. For this reason we develop a novel and previously unpublished component for lemmatization.

We represent lemmatization as a sequence-to-sequence translation problem, where the input is a word represented as a sequence of characters concatenated with a sequence of its part-of-speech and morphological tags, while the desired output is the corresponding lemma represented as a sequence of characters. Therefore we are training the system to translate the word form characters + morphological tags into the lemma characters, where each word is processed independently from it's sentence context. For example, input and output sequences for the English word *circles* as a

noun are:

```
INPUT: c i r c l e s UPOS=NOUN
       XPOS=NNS Number=Plur

OUTPUT: c i r c l e
```

As our approach can be seen similar to general machine translation problem, we are able to use any openly available machine translation toolkit and translation model implementations. Our current implementation is based on the Python version of the OpenNMT: Open-Source Toolkit for Neural Machine Translation (Klein et al., 2017). We use a deep attentional encoder-decoder network with 2 layered bidirectional LSTM encoder for reading the sequence of input characters + morphological tags and producing a sequence of encoded vectors. Our decoder is a 2 layered unidirectional LSTM with input feeding attention for generating the sequence of output characters based on the encoded representations. In input feeding attention (Luong et al., 2015) the previous attention weights are given as input in the next time step to inform the model about past alignment decisions and prevent the model to repeat the same output multiple times. We use beam search with beam size 5 during decoding.

As the lemmatizer does not see the actual sentence where a word appears, morphological tags are used in the input sequence to inform the system about the word's morpho-syntactic context. The tagger is naturally able to see the full sentence context and in most cases it should produce enough information for the lemmatizer to give it a possibility to lemmatize ambiguous words correctly based on the current context. During test time we run the lemmatizer as a final step in the parsing pipeline, i.e. after tagger and parser, so the lemmatizer runs on top of the predicted part-of-speech and morphological features. Adding the lemmatizer only after the tagger and parser (and not before like done in many pipelines) does not cause any degradation for the current pipeline as the tagger and parser by Dozat et al. (2017) do not use lemmas as features.

This method is inspired by the top systems from the CoNLL-SIGMORPHON 2017 Shared Task of Universal Morphological Reinflection (Cotterell et al., 2017), where the participants used encoder-decoder networks to generate inflected words from the lemma and given morphological tags (Kann and Schütze, 2017; Bergmanis et al., 2017). While the SIGMORPHON 2017 Shared Task was based on gold standard input features, to our knowledge we are the first ones to use similar techniques on reversed problem settings and to incorporate such lemmatizer into the full parsing pipeline to run on top of predicted morphological features.

4 Near-zero resource languages

There are nine very low resource languages: Breton, Faroese, Naija and Thai with no training data, and Armenian, Buryat, Kazakh, Kurmanji and Upper Sorbian with only a tiny training dataset. For the latter five treebanks with tiny training sample, we trained the tagger and parser in the standard manner, despite the tiny training set size. However, for four of these five languages (Armenian, Buryat, Kazakh and Kurmanji) we used Apertium morphological transducers (Tyers et al., 2010) to artificially extend the lemmatizer training data by including new words from the transducer not present in the original training data (methods are similar to those used with Breton and Faroese, for details see Section 4.1). Naija is parsed using the English-EWT models without any extra processing as it strongly resembles English language and at the same time lacks all resources. Breton, Faroese and Thai were each treated in a different manner described below.

4.1 Breton

Our approach to Breton was to first build a Breton POS and morphological tagger, and subsequently apply a delexicalized parser. To build the tagger, we selected 5000 random sentences from the Breton Wikipedia text dump and for each word looked up all applicable morphological analyzes in the Breton Apertium transducer converted into UD using a simple language-agnostic mapping from Apertium tags to UD tags. For words unknown to the transducer (59% of unique words), we assign all possible `UPOS+FEATS` strings produced by the transducer on the words it recognizes in the data. Then we decode the most likely sequence of morphological readings using a delexicalized 3-gram language model trained on the `UPOS+FEATS` sequences of English-EWT and French-GSD training data. Here we used the `lazy` decoder program[5] which is based on the KenLM language model estimation and querying system (Heafield, 2011). This procedure re-

[5] https://github.com/kpu/lazy

sults in 5000 sentences (96,304 tokens) of morphologically tagged Breton, which can be used to train the tagger in the usual manner. The syntactic parser was trained as delexicalized (FORM field replaced with underscore) on the English-EWT and French-GSD treebanks. The accuracy of UPOS and FEATS was 72% (3rd rank) and 56.6% (2nd rank) and LAS ranked 3rd with 31.8%. These ranks show our approach as competitive in the shared task, nevertheless the Uppsala team achieved some 14pp higher accuracies of UPOS and FEATS, clearly using a considerably better approach.

The Breton lemmatizer was trained using the same training data as used for the tagger, where for words recognized by the transducer the part-of-speech tag and morphological features are converted into UD with the language-agnostic mapping, and lemmas are used directly. Unknown words for transducer (i.e. those for which we are not able to get any lemma analysis) are simply skipped from the lemmatizer training. As the lemmatizer sees each word separately, skipping words and breaking the sentence context does not cause any problems. With this approach we achieved the 1st rank and accuracy of 77.6%, which is over 20pp better that the second best team.

To estimate the quality of our automatically produced training data for Breton tagging and lemmatization, we repeat the same procedure with the Breton test data[6], i.e. we use the combination of morphological transducer and language model as a direct tagger leaving out the part of training an actual tagger with the produced data as done in our original method. When evaluating these produced analyses against the gold standard, we get a direct measure of quality for this method. We measure three different scores: 1) Oracle full match of transducer readings converted to UD, where we measure how many tokens can receive a correct combination of UPOS and all morphological tags when taking into account all possible readings given by the transducer. For unknown words we include all combinations known from the transducer. This setting measures the best full match number achievable by the language model if it would predict everything perfectly. 2) Language model full match, i.e. how many tokens received a fully correct analysis when language model was used to pick one of the possible analyses. 3) Random choice full match, i.e. how many tokens received a fully correct analysis when one of the possible analyses was picked randomly. On Breton test set our oracle full match is 55.5%, language model full match 51.0% and random full match 46.2%. We can see that using a language model to pick analyses shifts the performance more closer to oracle full match than random full match, showing somewhat positive results for the language model decoding. Unfortunately when we tried to replicate the same experiment for other low-resource languages, we did not see the same positive signal. However, the biggest weakness of this method seems to be in the oracle full match which is only 55.5%. This means that the correct analysis cannot be found from the converted transducer output for almost half of the tokens. A probable reason for this is the simple language-agnostic mapping from Apertium tags to UD tags which is originally developed for the lemmatizer training and strove for high precision rather than high recall. Our development hypothesis was that missing a tag in lemmatizer's input likely does not tremendously harm the lemmatizer, so when developing the mapping we rather left some tags out than caused a potential erroneous conversion. However, when the same mapping is used here, missing one common tag (for example VerbForm=Fin) can cause great losses in full match evaluation.

[6]Using development data in these experiments would be more desirable, but unfortunately we don't have any Breton development data available.

4.2 Faroese

For Faroese the starting situation was similar to Breton but as the coverage of the Faroese Apertium tranducer was weak, we decided to take an another approach. This is because we feared that the decoder input would have too many gaps to fill in and therefore the quality of produced data would decrease. For that reason the Faroese tagger and parser was trained in the usual manner using pooled training sets of related Nordic languages: Danish-DDT, Swedish-Talbanken, and the three available Norwegian treebanks (Bokmaal, Nynorsk, NynorskLIA). The pre-trained embeddings were Faroese from the Facebook's embeddings dataset, filtered to only contain words which Faroese has in common with one of the languages used in training. However, the Faroese lemmatizer is trained directly from the transducer output by analyzing vocabulary extracted from the

Faroese Wikipedia and turning Apertium analyses into UD using the same tag mapping table as in the Breton. On UPOS tagging our system ranks only 10th, whereas on both morphological feature prediction and lemmatization, we rank 1st.

4.3 Thai

As there is no training data and no Apertium morphological transducer for Thai, we machine translated the English-EWT treebank word-for-word into Thai, and used the result as training data for the Thai segmenter, tagger and parser. Here we utilized the Marian neural machine translation framework (Junczys-Dowmunt et al., 2018) trained on the 6.1 million parallel Thai-English sentences in OPUS (Tiedemann, 2012). Since we did not have access to a Thai tokenizer and Thai language does not separate words with spaces, we forced the NMT system into character-level mode by inserting a space between all characters in a sentence (both on the source and the target side) and again removing those after translation. After training the translation system, the English-EWT treebank is translated one word at a time, creating a token and sentence segmented Thai version of the treebank. Later all occurrences of English dots and commas were replaced with whitespaces in the raw input text (and accordingly absence of *SpaceAfter=No* tags in CoNNL-U) as Thai uses whitespace rather than punctuation as pause character, and rest of the words were merged together in raw text by including *SpaceAfter=No* feature for each word not followed by dot or comma. This word-by-word translation and Thai word merging technique gives us the possibility to train a somewhat decent sentence and word segmenter without any training data for a language which does not use whitespaces to separate words or even sentences. Furthermore, all *the* words were removed as they have no Thai counterpart, lemmas were dropped, all matching morphological features between English and Thai were copied, HEAD indices were updated because of removing before mentioned tokens, non-existent dependency relations in Thai were mapped to similar existent ones, and finally enhanced dependency graphs were dropped. The tagger and parser were then trained normally using this training data. Training a lemmatizer is not needed as the Thai treebank does not include lemma annotation.

Our Thai segmentation achieves 1st rank and accuracy of 12.4% on sentence segmentation and 5th rank and accuracy of 43.2% on tokenization. On UPOS prediction we have accuracy of 27.6% and 4th rank, and our LAS is 6.9% and we rank 2nd, while the best team on Thai LAS, CUNI x-ling, achieves 13.7%. English is not a particularly natural choice for the source language of a Thai parser, with Chinese likely being a better candidate. We still chose English because we were unable to train a good Chinese-Thai MT system on the data provided in OPUS and the time pressure of the shared task prevented us from exploring other possibilities. Clearly, bad segmentation scores significantly affect other scores as well, and when the parser and tagger are evaluated on top of gold segmentation, our UPOS accuracy is 49.8% and LAS 20.4%. These numbers are clearly better than with predicted segmentation but still far off from typical supervised numbers.

5 Results

The overall results of our system are summarized in Table 3, showing the absolute performance, rank, and difference to the best system / next best system for all metrics on several treebank groups — big, small, low-resource and parallel UD (PUD). With respect to the three main metrics of the task, we ranked 2nd on LAS, 2nd on MLAS and 1st on BLEX, and received the highest aggregate ranking out of 26 teams, of which 21 submitted non-zero runs for all treebanks. For LAS, our high rank is clearly due to balanced performance across all treebank groups, as our ranks in the individual groups are 3rd, 6th, 4th and 6th, still giving a 2nd overall rank. A similar pattern can also be observed for MLAS. Our 1st overall rank on the BLEX metric is undoubtedly due to the good performance in lemmatization, on which our system achieves the 1st rank overall as well as in all corpus groups except the low-resourced languages. Altogether, it can be seen in the results table that the two main strengths of the system is 1) lemmatization and 2) tagging of small treebanks, and on any metric, the system ranks between 1st and 5th place across all corpora (*all* column in Table 3).

6 Software release

The full parsing pipeline is available at `https://turkunlp.github.com/Turku-neural-parser-pipeline`,

	All	*Big*	*PUD*	*Small*	*Low*
LAS	73.28 (-2.56 / 2)	81.85 (-2.52 / 3)	71.78 (-2.42 / 6)	64.48 (-5.05 / 4)	22.91 (-4.98 / 6)
MLAS	60.99 (-0.26 / 2)	71.27 (-1.40 / 3)	57.54 (-1.21 / 5)	47.63 (-1.61 / 2)	3.59 (-2.54 / 5)
BLEX	**66.09 (+0.76 / 1)**	**75.83 (+0.37 / 1)**	**63.25 (+0.91 / 1)**	53.54 (-1.35 / 2)	11.40 (-2.58 / 2)
UAS	77.97 (-2.54 / 4)	85.32 (-2.29 / 5)	75.58 (-2.84 / 6)	71.50 (-4.44 / 5)	34.51 (-4.72 / 6)
CLAS	69.40 (-2.96 / 2)	78.26 (-3.03 / 4)	67.65 (-2.21 / 5)	59.28 (-5.57 / 4)	18.15 (-4.03 / 6)
UPOS tagging	89.81 (-1.10 / 4)	95.41 (-0.82 / 6)	85.59 (-1.92 / 9)	91.93 (-0.91 / 3)	52.53 (-8.54 / 4)
XPOS tagging	86.17 (-0.50 / 3)	94.47 (-0.69 / 4)	55.68 (-0.30 / 2)	**90.51 (+0.50 / 1)**	43.43 (-11.3 / 17)
Morph. features	86.70 (-0.89 / 3)	93.82 (-0.32 / 3)	85.24 (-1.81 / 5)	**85.63 (+0.58 / 1)**	40.04 (-8.91 / 4)
All morph. tags	79.83(-0.47 / 2)	91.08 (-0.42 / 3)	51.60 (-0.30 / 2)	**82.02 (+1.17 / 1)**	17.58 (-8.33 / 19)
Lemmatization	**91.24 (+1.92 / 1)**	**96.08 (+0.83 / 1)**	**85.76 (+0.07 / 1)**	**91.02 (+1.02 / 1)**	61.61 (-2.81 / 3)
Sentence segmt.	83.03 (-0.84 / 5)	86.09 (-3.43 / 7–21)	75.53 (-0.51 / 3–17)	83.33 (-0.12 / 2–20)	66.23 (-1.27 / 2)
Word segmt.	97.42 (-0.76 / 5)	98.81 (-0.40 / 8–21)	92.61 (-1.96 / 7–19)	99.43 (+0.20 / 1–19)	89.10 (-4.28 / 5)
Tokenization	97.83 (-0.59 / 4)	99.24 (-0.27 / 6–21)	92.61 (-1.96 / 7–19)	99.57 (+0.01 / 1–18)	89.85 (-3.49 / 5)

Table 3: Results in every treebank group, shown as "absolute score (difference / rank)". For first rank, the difference to the next best system is shown, for other ranks we show the difference to the best ranking system, shared ranks are shown as a range.

together with all the trained models. We have ported the parser of Dozat et al. into Python3, and included other modifications such as the ability to parse a stream of input data without reloading the model. The pipeline has a modular structure, which allowed us to easily reconfigure the components for languages which needed a non-standard treatment. The pipeline software is documented, and we expect it to be comparatively easy to extend it with own components.

7 Conclusions

In this paper we presented the TurkuNLP entry at the *CoNLL 2018 UD Shared Task*. This year we focused on building an end-to-end pipeline system for segmentation, morphological tagging, syntactic parsing and lemmatization based on well-known components, and including our novel lemmatization approach. On BLEX evaluation, a metric including lemmatization and syntactic tree, we rank 1st, reflecting the state-of-the-art performance on lemmatization. On MLAS and LAS, metrics including morphological tagging and syntactic tree, and plain syntactic tree, we rank 2nd on both. All these components are wrapped into one simple parsing pipeline that carries out all four tasks with one command, and the pipeline is available for everyone together with all trained models.

Acknowledgments

We would like to thank Tim Dozat and rest of the Stanford team for making their parser open-source, as well as Milan Straka and rest of the Prague team for making UDPipe software and models open-source. This work was supported by Academy of Finland, Nokia Foundation and Google Digital News Innovation Fund. Computational resources were provided by CSC – IT Center for Science, Finland.

References

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, Vancouver, pages 31–39. http://www.aclweb.org/anthology/K17-2002.

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

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, Vancouver, pages 1–30. http://www.aclweb.org/anthology/K17-2001.

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

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings*

of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies pages 20–30.

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 ('UFAL), Faculty of Mathematics and Physics, Charles University. http://hdl.handle.net/11234/1-1989.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland, United Kingdom, pages 187–197. https://kheafield.com/papers/avenue/kenlm.pdf.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*. Melbourne, Australia. https://arxiv.org/abs/1804.00344.

Katharina Kann and Hinrich Schütze. 2017. The lmu system for the conll-sigmorphon 2017 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, Vancouver, pages 40–48. http://www.aclweb.org/anthology/K17-2003.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of the 55th annual meeting of the Association for Computational Linguistics (ACL'17)*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*. pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

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.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

Francis Tyers, Felipe Sánchez-Martínez, Sergio Ortiz-Rojas, and Mikel Forcada. 2010. Free/open-source resources in the apertium platform for machine translation research and development. *The Prague Bulletin of Mathematical Linguistics* 93:67–76.

Daniel Zeman, Filip Ginter, Jan Hajič, Joakim Nivre, Martin Popel, Milan Straka, and 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*. Association for Computational Linguistics, pages 1–20.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Brussels, Belgium, pages 1–20.