CEA LIST : Processing low-resource languages for CoNLL 2018 Shared Task

Elie Duthoo **Olivier Mesnard**
CEA, LIST, Laboratory of Vision and Content Engineering
elie.duthoo@protonmail.com
olivier.mesnard@cea.fr

Abstract

In this paper, we describe the first CEA LIST participation at the CoNLL 2018 shared task. The submitted system is based on the state of the art parser from CoNLL 2017, that has been improved by the addition of morphological features predictions and the integration of additional resources to provide accurate models for low-resource languages. Our approach ranked 5^{th} of 27 participants in MLAS for building morphology aware dependency trees, 2^{nd} for morphological features only, and 3^{rd} for tagging (UPOS) and parsing (LAS) low-resource languages.

1 Introduction

The CoNLL 2018 Shared Task (Zeman et al., 2018) is dedicated to developing dependency parsers on many languages, including low-resource languages. Our system uses the Stanford team parser[1] (Dozat et al., 2017) which was at the state of the art during the CoNLL 2017 shared task. We will refer to it as "stf parser" in this article. We also used UDPipe (Straka et al., 2016) for tokenization, sentence segmentation, word alignment, lemmas and XPOS tags. Our main purpose is not to propose a new parser system, our approach is mainly focused on the adaptation of existing systems to low-resources languages.
However, we also propose some improvements of the stf parser on multiple levels: (1) the training time is shorter and the models are more accurate for features and XPOS tags predictions (even if we didn't use the predicted XPOS tags in the final submission) (2) we studied the hyper-parameters in order to find the best configuration (3) we implemented an optimal tree construction instead of a greedy one, based on Stanford team recommendations in their 2017 paper.
We spent five man-month to provide these results, with two additional man-month dedicated to Breton corpus.
For low-resource languages, we based our approach on the following available data : OPUS corpus, Wikipedia data (Wiktionary) and word embeddings.

[1] https://github.com/tdozat/Parser-v2

2 Architecture

Our architecture mainly reused the proposed architecture of the 2017 state of the art system. Here is a brief reminder to explain what we used and how we used it. The model uses character embeddings, pre-trained word embeddings and post-trained word embeddings.

- Firstly, the raw text is processed by UDPipe.
- Then the stf tagger annotates the data with UPOS, XPOS and Features tags. Both the tagger and the parser use bi-LSTM layers on word embeddings. The tagger and the parser are separately trained models even if the models structures are really close.
- Finally, for each word, the parser concatenates one word level embedding (which is the sum of chars + pre-trained + trained embeddings) with one POS tag level embedding (UPOS + XPOS + features). POS tag embeddings are also learned during the training. We didn't use lemmas. The stf parser is graph-based.

We only used XPOS tags where they were contributing to the UPOS, XPOS, features trio. We evaluated this contribution by comparing the number of unique XPOS tags per corpus to the number of unique features. Some XPOS tags contains multiple tag information (which is almost always

Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 34–44

https://doi.org/10.18653/v1/K18-2003

the case for the features) and could thus contribute more to the word embedding. For almost all languages, the parser is based on three bi-LSTM layers (300neurons/layer) and the tagger on two bi-LSTM layers(200neurons/layer). The tagger ends with a 120 neurons lReLU layer. The parser ends with a 500 neurons lReLU layer for learning arc probabilities, and a 170 neurons lReLU layer for predicting the labels of the dependencies. For languages with less than 1500 sentences, we used a parser composed of only one bi-LSTM layer and smaller lReLU layers.
We only kept the best system evaluated on dev data for each language on each of the last two steps. Our measure for the tagger was the combination of UPOS and Features annotations (like Alltags without XPOS), if UDPipe was better than stf, we used UDPipe. Our measure for the parser was the LAS. For the evaluation, the predicted XPOS tags were removed from the parsed file and changed to the UDPipe ones, however the predicted XPOS were used by the parser when they were predicted by the tagger. We didn't evaluate the XPOS tags as we needed to sort the sub-XPOS tags (these sub-POS tags are sorted in alphabetical order for features which is really more convenient). The problem was only that we had to sort the sub-XPOS tags in order to have a score on them, but sorting them is not required to have them used in the parser.
A table has been produced to summarize the configuration (cf table 1).

3 Enhancements

3.1 Features

The initial architecture provided by Stanford team was restricted to predicting UPOS tags only, or UPOS tags and XPOS tags. We added a third type of tagger to predict UPOS tags, XPOS tags and morphological features at the same time.
As we said earlier, the tagger takes into account characters by using a LSTM on each character and using a linear attention layer on the LSTM outputs, so we did expect great results on features. However, POS tags are heterogeneously formatted: the UPOS tag represents only one information, but a feature tag generally contains many sub-tags, and XPOS tags may also contain sub-tags depending on the treebank. The original architecture didn't tackle this problem and was processing all tags as if they were unary tags. It means, for example, that for Latin-ITTB, the tagger ended with a lReLU layer that was choosing the best class between 3129 classes.
We added a lReLU layer for each sub-class at the end of the network so that the backpropagation only backpropagates error on a subclass level, thus being more accurate when building character embeddings. This architecture was extended to XPOS tags, as they may also contains subclasses. Finally the score that was used to stop the training and keep the best model was the "UPOS and Features" score, counting a word ok only if it had the exact UPOS tag and the exact features tags.

To sum up, for each sub-category (for example, the tense of a verb), a different lReLU layer determines if the word corresponds to one of the sub-category possible values (*none*, *present*, *future*, *past*..).

3.2 Hyperparameters

In order to understand the impact of the parameters on the architecture, we performed some random search optimization on hyperparameters, for a subset of languages (Russian, Hebrew, French, Uyghur, Vietnamese, Ancient Greek and Bulgarian) that we considered representative. We studied the following parameters: dropout on word embeddings, dropout on character embeddings, word embeddings merging strategy (whether to sum or concatenate the word embedding coming from characters and the pre/post trained word embeddings), case sensitivity, number of bi-LSTM layers, number of neurons for bi-LSTM, lReLU layers sizes, lReLU layers dropout, and some optimizer parameters like the learning rate.

- **number of bi-LSTM** The number of bi-LSTM layers deeply affects the score probably because of the vanishing gradient problem. Best scores were achieved for one to three bi-LSTM. With four layers, some models fail to converge and with five layers, no converging model is found.
- **dropout** For dropout on character embeddings, we found that is wasn't useful (initial value was 0.5 (50%) but optimal value was between 0 and 0.1). Other dropout did not have a clear influence.
- **learning rate** The learning rate was tuned based on the model (tagger / parser) as the optimal value wasn't the same depending on the task and

Lang	Tokenizer	Tagger	XPOS	Parser	Smaller neural network
af-afribooms	baseline	stf	used	stf	yes
br-keb	nl-alpino	stf	unused	stf	no
bxr-bdt	baseline	udp	unused	udp	no
en-lines	baseline	stf	used	stf	no
fo-oft	da-ddt	stf	unused	stf	no
fr-spoken	baseline	stf	unused	stf	yes
ga-idt	baseline	udp	unused	udp	yes
gl-ctg	baseline	stf	used	stf	no
gl-treegal	baseline	udp	unused	udp	no
hsb-ufal	baseline	udp	unused	udp	no
hu-szeged	baseline	stf	unused	stf	yes
hy-armtdp	baseline	udp	unused	stf	yes
kk-ktb	baseline	udp	unused	udp	no
kmr-mg	baseline	udp	unused	udp	no
ko-gsd	baseline	stf	used	stf	no
ko-kaist	baseline	stf	used	stf	no
la-ittb	baseline	stf	used	stf	no
la-perseus	baseline	udp	unused	udp	no
nl-alpino	baseline	stf	used	stf	no
nl-lassysmall	baseline	stf	used	stf	no
no-nynorsklia	baseline	udp	unused	no-nynorsk	no
pcm-nsc	en-lines	stf	unused	stf	no
ru-taiga	baseline	udp	unused	ru-syntagrus	no
sl-sst	baseline	udp	unused	udp	no
sme-giella	baseline	udp	unused	udp	no
th-pud	own	stf	used	stf	no
vi-vtb	baseline	stf	used	stf	yes
zh-gsd	baseline	stf	used	stf	no
all others	baseline	stf	unused	stf	no
Other models used					
cs-pud & cs-pdt, sv-pud & sv-lines, fi-pud & fi-tdt, ja-modern & ja-gsd (udp), en-pud & en-gum					
pcm-nsc	en-lines (stf without post-trained embeddings)				

Table 1: Summary table of specific configurations per languages. udp = UDPipe architecture

both tasks didn't have the same optimal architecture.

- **other parameters** Other parameters did not have a clear empirical influence on the score, such as the number of neurons on each layer (from 50 to 500) and other parameters of the optimizer.

3.3 Stopping criteria

Depending on the model (tagger/parser), the stopping criteria was based on UPOSandFeats score or LAS. The first thing we did with the parser was to improve the training time. The previous algorithm stopped the training only if the model didn't improve at all for 5000 iterations (which could correspond to 2 hours on our environment), or after a maximum number of iterations. This means that the training would go on even for an improvement of 0.01% each hour. In practice, with this configuration, the longest model took 24 hours to train. We added two stopping criteria:

- The first is the time, the training time shouldn't exceed 2 hours. The training time depends on the architecture so we chose the value depending on our experiments
- The second is the average progression of the system per hour during the last 20 validations (we set one validation each 120 iteration). If the system was under 0.1% per hour, we stopped the training.

With these criteria, almost all models are converging really fast (in 30 minutes they were already close by 1% to their best score).

3.4 Optimal directed spanning tree

The graph-based model parser predicts a probability of parent for each word of the sentence. The default algorithm in the stf parser is greedy but the Stanford team recommended to implement an op-

timal algorithm to provide non-projective dependency trees as a lot of treebanks contains at least one non-projective tree.
In fact, 99% of all trees are projectives (non-weighted average across all proportions in all treebanks). But building the optimal non-projective spanning tree based on probabilities should also help in building projective trees (as projective trees are a subset of non-projective trees), and some treebanks contain a lot more of non-projective trees (10% for Ancient Greek Perseus).
We implemented the Chu-Liu-Edmonds algorithm (Edmonds, 1967) and evaluated the result. We didn't notice a great improvement (less than 0.1%), we suppose that the probabilities were high enough so that a greedy algorithm doesn't have troubles predicting the optimal tree by always choosing the highest probability.
Empirically, the stf parser does build more nonprojective trees than in the training set, one could consider implementing the Eisner algorithm (Eisner, 1996) and make a compromise between Edmonds and Eisner algorithms based on the probabilities in each tree and on the non-projective proportion wished.

3.5 Including an embedding from the tagger in the parser

In the stf parser, the tagger and the parser are separated. The parser builds its own UPOS tag embeddings but do not take into account the probabilities of each tag proposed by the tagger. The fact that one word has one tag instead of another could be subjective in some circumstances and as the tagger doesn't get 100% in UPOS tag prediction, the parser should get the information that the tag it is reading is just a choice over many others, based on probabilities.
For some languages like French, the tagger could confuse verbs and auxiliaries. For these tags, the probability could be something like 70% verb and 30% auxiliary, and the tagger will label the word as a "verb". Making a mistake on which word is a verb and which one is an auxiliary affects the dependency tree. We therefore extracted these probabilities from the tagger to introduce them in the parser.
This experiment didn't show a clear improvement in the results. We think that the parser has trouble learning from these probabilities because they are almost always at one (the tagger rarely makes mistakes). Maybe one could improve the system by not adding the probabilities directly but the output of the layer before the softmax. This improvement was not included in the final submission.

4 Cross-lingual transfer for low-resource languages

The rest of our work was focused on cross-lingual transfer methods and data mining to address the problem of low-resource languages. The main idea of our models is to build an artificial treebank on which we could learn a tagger / parser. As exposed by (Tiedemann and Agić, 2016), many techniques may be used to build artificial treebanks. They identified mainly 1) model transfer (for example, building a delexicalized parser from a rich-resource language and using it directly on a target language corpus annotated beforehand with POS tags), 2) direct annotation projection (relying on word alignments of a bitext to project POS annotation), 3) treebank translation and 4) cross-lingual word embeddings. The use of partial annotations with dictionary may complete these four techniques. We relied on quality of available data to choose the best suited technique. Our analysis started on all low-resource languages but we applied our process to only 3 languages: Breton, Faroese and Thai because of the lack of valuable data for others[2]. Extending the capacities of our models is certainly doable but a lot of regularization on data should be done. We present a summary of our choices in table 2.

4.1 Breton

Breton treebank has been built with direct annotation projection (Tiedemann and Agić, 2016). There were neither training nor evaluation treebank for Breton. Breton is member of Celtic language family but no large treebank exists in this family (Irish_IDT is a 1020 only sentences corpus). We decided to use the most valuable and available resources for Breton: Wiktionary, word embeddings from Facebook[3], combined as in Wisniewski et al. (2014) with cross-lingual projection from a larger parallel corpus to build our own artificial learning corpus.
We selected OfisPublik from OPUS (Tiedemann, 2009) site as parallel corpus for cross lingual

[2]We also had results on Upper Sorbian but failed to submit the model.
[3]https://git.io/fbjDv

Role	Breton	Thai	Faroese
Tokenization	model transfer	dictionary	model transfer
POS	annotation projection+Wiktionary	annotation projection+Wiktionary	model transfer (*)
Dependencies	annotation projection	model transfer (*)	model transfer (*)

Table 2: choices made to build our artificial treebanks for low-resource languages
(*) with cross-lingual embeddings

transfer. This corpus is composed of more than 60.000 sentences dumped from a bilingual institutional site about Breton language[4].
We proceeded as follow:

Statistics on parse tree: We computed some statistics on Irish treebank: the counts of occurrences of all types of arc, indexed by triples (head UPOS, dependency UPOS, arc label) to setup a prior distribution probability.

Lexicon: We built a lexicon of form-category from the most recent brwiktionary dump (20/05/2018). We obtain 26650 entries with mostly one category. When there are more than one category, we preserved the order, assessing that the most frequent category is the first mentioned in Wiktionary. Using links to other Wiktionaries we manually translated the 53 categories into UPOS tag.

Parallel corpus: We filtered the parallel corpus and excluded some sentences: those with less than 3 words (br or fr), those with more than 30 words (br or fr) and those whose ratio of size of original sentence upon translated sentence is not in the range [1/3,3].

Tokenization: We used UDPipe with Dutch model to tokenize the Breton side of the parallel corpus. We chose Dutch because it preserves "c'h" sequence of character which is the transcript of a very frequent consonant in Breton.

Word Alignment: We built a word alignment of the tokenized parallel corpus with efmaral[5]. We used forward and reverse alignment and combined them with *atools* from fastalign[6] with *grow-diag-final-and* mode.

Annotation: We used UDPipe to annotate the French side corpus with baseline UDPipe model for French.

POS tagging: We have performed UPOS annotation within four steps.

1. Use our lexicon to annotate supposedly non ambiguous tokens with word type: when the form of a token has an entry with only one category, the value is used as UPOS.
2. Use our lexicon to annotate ambiguous tokens with hypotheses of word type : when a form has an entry with more than one category, the set of values is associated to the token within the *upos_lexicon* attribute.
3. Used bre2fre alignment to collect hypotheses of token type: when a token is in the bre2fre mapping, we collect the UPOS of all tokens in French side within the *upos_alignment* attribute
4. Intersect *upos_alignment* and *upos_lexicon* to select most likely UPOS.

At the end of the process, sentences without complete POS tagging are discarded.

Lemma: We have not built any specific solution for lemmatisation and we reproduce the form as value for lemma.

Features: We have projected features from the French side. Because there is no features in our lexicon, features comes only from cross-lingual projection. At the end of first pass we collect all features annotation and we add this data to our lexicon. Then, we use this enhanced lexicon as source to annotate tokens with features when this information exists in the lexicon.

Head and label: We walk through French dependency tree starting from root node, and use alignment fre2bre to project dependency annotation (arc and label) when structures of French and Breton seem to be the same. We initialize a stack with the pair (root of French tree, root of Breton tree), the second member is chosen as the word aligned with root of French tree.
As long as the stack is not empty:

- pop a pair (*fre_head*, *bre_head*)
- get *fre_child*ren from *fre_head* and get UPOS from each *fre_child*
- for each *bre_node* in fre2bre alignment of each *fre_child*
- if many conditions are met (there is only one *bre_node* aligned with *fre_child*, *fre_child* and *bre_node* have same POS, there is only one word aligned with *fre_head*) we simply project head

[4] http://www.fr.brezhoneg.bzh
[5] https://github.com/robertostling/efmaral
[6] https://github.com/clab/fast_align

and label, i.e. head of *bre_node* is *bre_head*, label is label of arc (*fre_child*, *fre_head*).

- when such conditions are not met, we create hypothesis of dependency with most likely pair of nodes: *bre_node* with *bre_head* but also with all siblings of *bre_node*. We use statistics to associate a probability to each of these hypotheses.
- add to stack (*fre_child*, *bre_node*)

Finally, we build the parse tree with Chu-Edmonds algorithm from the set of hypotheses.

Learning and prediction: We use this artificial treebank to learn the UDPipe tagger and the stf parser. We use the UDPipe tokenizer with Dutch model to preprocess the raw test text.

Evaluation: Because no evaluation data exists, we relied on another language pair (French-German) to experiment the cross-lingual projection process. We build a lexicon from dewiktionary with a mapping from Wiktionary tags to UPOS tags. We select the 15.000 most frequent forms in German based on word embeddings order. We use PUD_French and PUD_German treebanks as parallel corpus. We get a F1 measure of 72% on UPOS on and 28% on LAS which let us hope similar results with French-Breton pair. These figures are not far from our official results 75% in UPOS and 38% in LAS.

4.2 Faroese, Thai, and all other low-resource languages

We started by exploring all the data we had available for the shared task: Wikipedia, word embeddings and parallel corpus, to build a low-resource language strategy.

We used cross-lingual word representations for building lexicalized taggers and parsers through model transfer and annotation projection for tag disambiguation.

4.2.1 Tokenizing Thai

The Thai language agglutinates a lot of words so it can't be tokenized with spaces. Dictionary Based Longest Matching yielded good enough results (Haruechaiyasak et al., 2008), so we built a dictionary for Thai based on all words from the embeddings and the Wiktionary. Then we tokenized the opus Thai corpus with dictionary, and learned an UDPipe model on it to facilitate the integration of the model in the final submission.

However, we didn't know how to define the end of each sentence in Thai, knowing that UDPipe won't define an end based on words or on sentences lengths. We tried to build our own tokenizer based on word embeddings, sentence lengths, word lengths and eventually a dictionary to improve the "only character embedding based" model. Even if we lacked the time to finalize this contribution, we did manage to reach a f1 score close to UDPipe with random forest classifier on English LinES corpus (-0.2% tokens, -5% sentences). The tokenization score for Thai is 64.17%.

4.2.2 Building word alignment on sentences and a bilingual dictionary with OPUS

We used efmaral (Östling and Tiedemann, 2016) to get word alignment on OPUS. As we needed to be extremely confident on the built alignments, the null-prior parameter was set to 0.95 (it doesn't stop the word aligner from making mistakes on some words).

The dictionary was built based on aligned words.

4.2.3 Tagging Thai

Three sources of information are available for tagging without annotated data:

- OPUS : tagging a rich-resource language and transferring tags to the low-resource language
- Embeddings alignment : aligning embeddings from one embedding cloud to another. Then using the source tagger on the aligned embeddings
- Data mining : parsing the Wiktionary to find the possible tags of each word

We can combine these methods to do disambiguation. The Thai language was tagged by using the same method as Breton. We extracted a list of each possible tags (UPOS) from the Wiktionary. Then we used annotated parallel corpus from Korean to disambiguate Thai tags. We weren't aware of a close enough language to Thai so we were only able to transfer from a distant language.

4.2.4 Embeddings alignment : tagging and parsing Faroese

For Faroese, the Wiktionary data didn't seem to be workable (no UPOS tag / not enough words) and no OPUS data were available. We thus used the only available source of informations : embeddings. We built a supervised cross-lingual embedding mapper that could work in an unsupervised way by using similar tokens between languages (when languages are close enough). The mapper is not limited by the number of word or the size of

the embedding contrary to MUSE (Lample et al., 2017). We found out that the HIT team had a similar approach in 2015 (Guo et al., 2015).
The Faroese language is close to three other languages : Danish, Norwegian and Swedish. The strategy was to build a bilingual tagger and a bilingual lexicalized parser for Swedish-Norwegian, evaluate our approach on Danish and export it on Faroese. All embeddings (Swedish, Danish and Faroese) were aligned in the Norwegian embedding cloud. The architecture used was the Stanford one, we only used gold and aligned embeddings (pretrained) and character embeddings (no post-trained embeddings as they are highly language dependent and based on word's form and not word's embedding position in the cloud).

The algorithm of the mapper is described in Algorithm 1: it uses the built bilingual dictionary (1,n) to (1,n) and all tokens of common form in embeddings as input, as well as the embeddings of the two languages.
Then, each embedding of the source language is aligned with the embeddings of the target language. The general case is the 1 to n relation, for which the source token is assigned an average of each of the n tokens in the target language. So if we have a 1 to 1 relation, the source token embedding is equal to the target token embedding. The dictionary was filtered by removing entries with cardinality above 4.
For words that are not in the dictionary, we use the n-closest embeddings in the source cloud which are in the dictionary. Each of these tokens has an alignment in the target cloud, so we do a weighted average of their aligned embeddings in the target cloud based on their distance in the source cloud to find the embedding of the token which is not in the dictionary.
This approach could be enhanced by taking into account the direction to go from each of the word in the source cloud to the target cloud, and averaging these informations to locate the unknown point in the target cloud (algorithm 1).

At this point we have cross-lingual embeddings. Building a bilingual corpus should help to reduce the bias of a monolingual model as well as learning with both original (Norwegian) and potentially biased aligned (Swedish) embeddings. We built a corpus based on UD ones (Nivre et al., 2017), merging the same number of sentences from Norwegian and Swedish into one file.We then trained a bilingual tagger and parser on this corpus without post-trained embeddings and evaluated it (cf table 3). SVNO : Swedish+Norwegian. DA : Danish. FO : Faroese.
We then used this bilingual model on Faroese Wikipedia sentences to have a treebank, annotating it by using aligned Faroese embeddings, and we trained a Faroese model on this treebank with original embeddings, hoping it could fix some inconsistencies coming from the alignment as the final model will use regularizations on original Facebook embeddings. In the end, we have a Faroese tagger and a Faroese parser.

Data: Source embeddings;
target embeddings;
dictionaries (opus + similar words)
Result: source aligned with target
init: define n closest words parameter;
for *each source embedding s of token ts* **do**
  let e be the embedding of ts in target;
  if *ts in dictionary* **then**
    ss = target words for ts in dic;
    es = embeddings of ss;
    e = avg(es);
  else
    cs = closest n word embeddings to s which are in dic;
    es = embeddings of the n closest word in target;
    e = weighted_avg(es, dist(cs,s));

Algorithm 1: Close-points embedding alignment algorithm; the n parameter was set to 5

4.2.5 Other languages

The method described in the previous section has been tested on other languages but did not produce results significantly better than the baseline. Not counting Thai, Faroese and Breton, there are 6 low-resource languages:

- For Buryat and Kurmanji we didn't find a close enough language to do unsupervised embedding alignment as no parallel corpus / Wiktionary were available.
- For Kazakh, we tried to build a Turkish-Russian model, ending with around 50% in UPOS.
- For Armenian, we tried a Persian-Greek model but we had troubles handling the character embeddings (Armenian characters aren't the same), ending with 23% in UPOS (and if you

Scores	SVNO score	No-transfer score
UPOS SV	93%	96%
UPOS DA	74%	97%
Feats SV	85%	92%
Feats DA	50%	97%
LAS SV	74%	83%
LAS DA	48%	84%
LAS SVNO	85%	
UPOS FO	64%	
Feats FO	34%	
LAS FO	47%	

Table 3: SVNO models results, compared to stf model trained on monolingual gold data.

don't include character embeddings, you can't distinguish some proper noun from numbers).

- For Naija, no data (OPUS/embedding/Wiktionary) was found, so we used English models.
- Finally for Upper Sorbian we had better results with a Polish-Czech model, ending with 75% UPOS and 37% LAS (evaluated on the 23 available sentences as we don't need training sentences), but we lacked of time to upload the model and finalize our contribution to the task.
  We tried to use embeddings alignment from MUSE on Upper Sorbian, but it didn't provide results as there wasn't enough Upper Sorbian embeddings.

5 Training

For the final submission, we trained around 190 models (including bilingual ones) which took 3 days on 3 GPU NVIDIA 1080 Ti. We did this full training 4 times, each time changing a parameter: whether to include language specific dependency annotation in the training, case-sensitive parser or to change the tokenization used (reference tokenization or UDPipe tokenization as validation set). The submitted result was the best model out of the four runs.

6 Results

We'll only detail some results. A summary table for low-resource languages we worked on has been produced (cf table 4).
We got some good ranks on the total average, but these scores were deeply affected by our rank in low-resource languages. For example, getting +64 points in Thai (with just a dictionary approach based on known words) is the same as getting 0.8 points on each other treebanks which is far more harder.
Still, we ranked 2^{nd} in morphological features, showing that the stf architecture was able to handle morphological informations.
We ranked 2^{nd} in tokenization (low-resource and global) because of our work on the Thai language (the 5 teams that got more than the baseline in Thai ended up in the top 5 on the global tokenization score). However our score for Thai tokenization (64%) is far more bellow other scores for tokenizing Thai in literature, based on other corpus which weren't available for this shared task.
We also ranked 3rd in UPOS (low-resource and global). We think that learning morphological features during the training help a model to get better results in UPOS as it could use the embeddings learned with morphological features to predict UPOS tags as well.
For big treebanks, we ranked 5^{th} in UPOS probably because we took the SOTA and improved it. We also stand below TurkuNLP for morphological features on big treebanks (93.68% for 93.82%), they also used the biLSTM architecture but they inferred tags without sub-categories.
Finally we ranked 5^{th} in MLAS which seems to be the more representative score overall as it evaluates morphology and dependency as a whole, and it doesn't give much importance to low-resource languages because a cumulative low score in UPOS and LAS results in a close to 0 MLAS.
A summary table can be found at the end of the article.

7 Conclusion

Our multilingual model seems to produce great results by enabling the production of a lexicalized parser for low-resource languages. This could

	Tokens			UPOS			LAS		
Lang	**Base**	**Our**	**Best**	**Base**	**Our**	**Best**	**Base**	**Our**	**Best**
Thai	8.56	64.17	69.93	5.86	31.46	39.42	0.70	0.47	13.70
Faroese	99.51	99.03	99.51	44.66	63.53	65.54	25.19	47.17	49.43
Breton	92.26	92.50	94.49	30.74	75.39	85.01	10.25	**38.64**	38.64

Table 4: Summary score table for 3 low-resource languages

open the road to an universal parser (as long as we have embeddings and a dictionary for a language) with improved performance over the mixed model provided for the shared task.

However, multiple steps are required in order to achieve a unique model. First, one should merge the tagger and the parser into one unique model, by learning the parser with representation learned by the tagger, and by eventually allowing the tagger to complete POS tags based on dependency learned embeddings.

Our transfer methods should be evaluated more carefully. Because we didn't have much time, we didn't find a way to evaluate our embeddings mapper in an other way than through the UPOS and dependency scores of our bilingual models.

Multiple strategies are still available from our work: how to select the best languages for transfer? How much languages should be used? How does these parameters change from one low-resource language to another ? etc.

Finally, to build an universal parser, we should distinguish multiple embeddings. The post-trained embeddings should be divided in two categories, the universal post-trained embeddings which changes the pretrained embeddings values based on their position into the embeddings cloud; and the language specific post-trained embeddings which should be based on the form and the language of the token. Including pretrained character embeddings and post-trained character embeddings could also help for languages with specific characters, for which we could at least map punctuations and numbers characters for tagging.

References

Timothy Dozat, Peng Qi, and Christoper Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task pages 20–30.

Jack Edmonds. 1967. Optimum branching. In *Journal of Research of the national Bureau of Standards B71(4)*. pages 233–240.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*. http://aclweb.org/anthology/C96-1058.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1234–1244. https://doi.org/10.3115/v1/P15-1119.

Choochart Haruechaiyasak, Sarawoot Kongyoung, and Matthew Dailey. 2008. A comparative study on thai word segmentation approaches 1:125 – 128.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043* .

Joakim Nivre et al. 2017. 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.

Robert Östling and Jörg Tiedemann. 2016. Efficient word alignment with Markov Chain Monte Carlo. *Prague Bulletin of Mathematical Linguistics* 106:125–146. http://ufal.mff.cuni.cz/pbml/106/art-ostling-tiedemann.pdf.

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. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria, volume V, pages 237–248.

Jörg Tiedemann and Zeljko Agić. 2016. Synthetic treebanking for cross-lingual dependency parsing. *Journal of Artificial Intelligence Research* 55:209–248.

Guillaume Wisniewski, Nicolas Pcheux, Souhir Gahbiche-braham, François Yvon, and Université Paris Sud. 2014. Cross-lingual part-of-speech tagging through ambiguous learning. In *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 1779–1785.

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.

	tokens	upos	ufeats	las	mlas
af_afribooms	99.75 (5)	97.24 (7)	96.91 (3)	84.16 (7)	73.15 (4)
ar_padt	99.98 (2)	90.35 (7)	86.93 (6)	71.84 (8)	62.17 (5)
bg_btb	99.92 (4)	98.72 (4)	97.38 (4)	89.59 (5)	82.55 (3)
br_keb	92.5 (2)	75.39 (2)	43.55 (3)	38.64 (1)	4.15 (3)
bxr_bdt	97.07 (4)	41.66 (8)	38.34 (3)	12.61 (11)	2.09 (5)
ca_ancora	99.97 (5)	98.49 (7)	97.75 (7)	90.22 (6)	82.09 (6)
cs_cac	99.97 (7)	98.83 (9)	93.8 (6)	89.55 (10)	79.92 (6)
cs_fictree	99.97 (6)	98.4 (6)	95.57 (4)	91.22 (4)	81.87 (3)
cs_pdt	99.93 (5)	98.44 (12)	92.75 (8)	89.06 (11)	78.63 (10)
cs_pud	99.28 (4)	96.86 (4)	90.43 (7)	84.89 (3)	71.65 (7)
cu_proiel	100 (1)	96.09 (3)	89.98 (1)	73.13 (5)	61.79 (4)
da_ddt	99.87 (4)	97.38 (4)	96.7 (5)	83.69 (3)	75.27 (4)
de_gsd	99.58 (4)	93.42 (10)	89.3 (4)	78.19 (4)	56.18 (4)
el_gdt	99.86 (4)	97.64 (4)	94.56 (2)	88.18 (6)	76.44 (3)
en_ewt	99.03 (5)	95.01 (7)	95.55 (6)	82.88 (4)	74.46 (4)
en_gum	99.75 (3)	95.07 (7)	95.97 (4)	81.47 (6)	70.59 (5)
en_lines	99.95 (3)	96.47 (8)	96.49 (5)	78.61 (7)	70.05 (8)
en_pud	98.79 (24)	93.28 (24)	95.27 (3)	79.84 (15)	69.97 (13)
es_ancora	99.97 (4)	98.61 (7)	98.08 (4)	89.58 (5)	82.33 (5)
et_edt	99.91 (5)	97 (5)	95.19 (4)	83.52 (4)	75.87 (4)
eu_bdt	99.96 (6)	95.97 (5)	92.19 (3)	83.13 (2)	71.7 (2)
fa_seraji	100 (2)	97.05 (6)	97.13 (3)	86.18 (2)	80.38 (5)
fi_ftb	100 (2)	95.83 (6)	96.23 (4)	87.14 (3)	79.09 (2)
fi_pud	99.63 (4)	97.32 (5)	96.29 (6)	88.59 (5)	82.42 (3)
fi_tdt	99.69 (4)	96.53 (7)	94.78 (6)	85.99 (4)	78.59 (5)
fo_oft	99.03 (23)	63.53 (2)	33.52 (5)	47.17 (2)	0.8 (2)
fr_gsd	99.66 (7)	96.1 (9)	95.85 (4)	85.03 (8)	76.41 (8)
fr_sequoia	99.79 (5)	97.4 (9)	96.77 (6)	87.66 (7)	80.2 (4)
fr_spoken	100 (2)	95.91 (6)	100 (2)	69.83 (9)	57.88 (8)
fro_srcmf	100 (2)	95.87 (3)	97.59 (2)	86.78 (3)	79.62 (2)
ga_idt	99.3 (5)	89.21 (9)	78.79 (5)	62.93 (14)	37.66 (8)
gl_ctg	99.84 (4)	96.98 (3)	99.01 (4)	81.6 (5)	69.28 (5)
gl_treegal	99.69 (2)	91.09 (10)	89.59 (5)	66.16 (12)	49.13 (11)
got_proiel	100 (2)	95.71 (3)	88.87 (3)	68.32 (4)	54.8 (3)
grc_perseus	99.96 (5)	91.94 (5)	89.71 (6)	73.41 (4)	52.14 (5)
grc_proiel	100 (2)	96.87 (7)	91.33 (4)	75.02 (6)	58.36 (6)
he_htb	99.98 (2)	82.6 (4)	80.82 (4)	63.66 (6)	51.3 (5)
hi_hdtb	100 (2)	97.38 (7)	93.56 (4)	91.72 (3)	77.59 (4)
hr_set	99.92 (4)	97.8 (7)	89.81 (7)	86.18 (5)	70.11 (6)
hsb_ufal	98.6 (3)	65.75 (6)	49.8 (2)	23.64 (16)	3.55 (14)
hu_szeged	99.81 (3)	95.4 (3)	92.75 (1)	76.96 (6)	65.68 (4)
hy_armtdp	97.21 (4)	65.4 (8)	57.07 (2)	28.41 (7)	7.58 (7)
id_gsd	100 (2)	93.94 (1)	95.35 (6)	78.84 (3)	67.81 (3)
it_isdt	99.75 (5)	97.91 (6)	97.61 (3)	90.01 (7)	82.44 (5)
it_postwita	99.73 (3)	95.9 (4)	96 (3)	71.77 (11)	59.12 (7)
ja_gsd	90.46 (6)	88.9 (6)	90.45 (6)	74.55 (8)	61.74 (9)
ja_modern	65.98 (5)	48.44 (8)	64.11 (9)	22.71 (9)	8.1 (9)
kk_ktb	93.11 (6)	48.94 (11)	46.86 (4)	24.21 (4)	7.62 (5)
kmr_mg	94.33 (4)	59.31 (6)	48.39 (2)	23.92 (9)	5.47 (7)
ko_gsd	99.81 (6)	96.12 (4)	99.62 (5)	83.61 (3)	79.21 (3)
ko_kaist	100 (2)	95.37 (4)	100 (2)	86.41 (3)	80.48 (3)
la_ittb	99.94 (5)	97.82 (13)	95.93 (4)	85.17 (7)	77.97 (5)
la_perseus	100 (2)	83.34 (11)	72.07 (8)	47.61 (12)	30.16 (11)
la_proiel	99.99 (5)	96.69 (3)	90.7 (4)	71.07 (4)	58.79 (3)
lv_lvtb	99.4 (6)	95.17 (2)	91.29 (3)	80.29 (5)	67.27 (2)
nl_alpino	99.83 (5)	95.84 (7)	95.66 (7)	85.79 (7)	73.38 (6)
nl_lassysmall	99.82 (6)	95.93 (7)	95.59 (5)	82.17 (7)	70.9 (5)
no_bokmaal	99.78 (6)	97.85 (6)	96.19 (5)	89.73 (4)	81.96 (5)
no_nynorsk	99.93 (5)	97.72 (4)	96.3 (4)	88.97 (5)	80.55 (5)
no_nynorsklia	99.99 (2)	85.15 (12)	86.54 (6)	56.87 (10)	41.73 (9)
pcm_nsc	99.71 (1)	57.21 (2)	43.09 (2)	16.06 (9)	2.35 (20)
pl_lfg	99.86 (7)	98.54 (4)	94.67 (6)	94.62 (3)	86.26 (4)
pl_sz	99.99 (3)	98.05 (4)	92.24 (4)	91.31 (4)	80.44 (3)
pt_bosque	99.71 (2)	96.61 (3)	95.85 (2)	87.72 (2)	75.72 (2)
ro_rrt	99.67 (6)	97.47 (5)	96.75 (6)	85.9 (4)	77.7 (4)
ru_syntagrus	99.6 (7)	98.2 (9)	95.69 (6)	89.96 (11)	83.27 (6)
ru_taiga	98.14 (2)	86.53 (11)	76.01 (7)	63.85 (7)	40.9 (8)
sk_snk	100 (2)	96.61 (4)	90.89 (2)	86.38 (4)	73.44 (2)
sl_ssj	98.29 (6)	96.85 (5)	93.11 (4)	86.72 (6)	78.65 (2)
sl_sst	100 (2)	88.5 (12)	80.15 (7)	46.95 (13)	34.19 (12)
sme_giella	99.84 (3)	87.69 (7)	82.41 (5)	56.98 (14)	46.05 (10)
sr_set	99.97 (2)	98.04 (3)	93.63 (5)	87.92 (5)	76.95 (4)
sv_lines	99.96 (3)	96.74 (2)	89.18 (5)	81.46 (4)	65.84 (5)
sv_pud	98.52 (6)	92.98 (9)	73.32 (21)	76.23 (11)	42.8 (13)
sv_talbanken	99.78 (7)	97.5 (3)	96.48 (5)	85.69 (6)	78.19 (5)
th_pud	64.17 (2)	31.46 (3)	60.22 (2)	0.47 (17)	0.16 (5)
tr_imst	99.86 (3)	93.42 (5)	91.94 (1)	63.78 (4)	55 (3)
ug_udt	99.22 (7)	89.43 (1)	87.07 (2)	62.75 (8)	43.82 (4)
uk_iu	99.67 (5)	97.03 (5)	90.71 (3)	83.64 (6)	71.12 (4)
ur_udtb	100 (2)	94.12 (4)	83.53 (3)	81.89 (4)	56.85 (3)
vi_vtb	84.26 (7)	77.38 (6)	83.99 (6)	44.35 (6)	37.98 (5)
zh_gsd	89.55 (7)	85.06 (9)	88.57 (8)	65.34 (9)	55.2 (8)

Table 5: Summary table for main scores. Green: top 1, yellow: top 3, orange: top 5