SEx BiST: A Multi-Source Trainable Parser with Deep Contextualized Lexical Representations

Kyungtae Lim, Cheoneum Park, Changki Lee, Thierry Poibeau

SEx BiST: A Multi-Source Trainable Parser with Deep Contextualized Lexical Representations

KyungTae Lim[*], Cheoneum Park[+], Changki Lee[+] and Thierry Poibeau[*]
[*]LATTICE (CNRS & ENS / PSL & Université Sorbonne nouvelle / USPC)
[+]Department of Computer Science, Kangwon National University
{kyungtae.lim, thierry.poibeau}@ens.fr,
{parkce, leeck}@kangwon.ac.kr

Abstract

We describe the SEx BiST parser (Semantically EXtended Bi-LSTM parser) developed at Lattice for the CoNLL 2018 Shared Task (*Multilingual Parsing from Raw Text to Universal Dependencies*). The main characteristic of our work is the encoding of three different modes of contextual information for parsing: (*i*) Treebank feature representations, (*ii*) Multilingual word representations, (*iii*) ELMo representations obtained via unsupervised learning from external resources. Our parser performed well in the official end-to-end evaluation (73.02 LAS – 4th/26 teams, and 78.72 UAS – 2nd/26); remarkably, we achieved the best UAS scores on all the English corpora by applying the three suggested feature representations. Finally, we were also ranked 1st at the optional event extraction task, part of the 2018 Extrinsic Parser Evaluation campaign.

1 Introduction

Feature representation methods are an essential element for neural dependency parsing. Methods such as Feed Forward Neural Network (FFN) (Chen and Manning, 2014) or LSTM-based word representations (Kiperwasser and Goldberg, 2016; Ballesteros et al., 2016) have been proposed to provide fine-grained token representations, and these methods provide state of the art performance. However, learning efficient feature representations is still challenging, especially for under-resourced languages.

One way to cope with the lack of training data is a multilingual approach, which makes it possible to use different corpora in different languages as training data. In most cases, for instance in the CoNLL 2017 shared task (Zeman et al., 2017), the teams that have adopted this approach used a multilingual delexicalized parser (i.e. a multi-source parser trained without taking into account lexical features). However, it is evident that delexicalized parsing cannot capture contextual features that depend on the meaning of words within the sentence.

Following previous proposals promoting a model-transfer approach with lexicalized feature representations (Guo et al., 2016; Ammar et al., 2016; Lim and Poibeau, 2017), we have developed the SEx BiST parser (Semantically EXtended Bi-LSTM parser), a multi-source trainable parser using three different contextualized lexical representations:

- ***Corpus representation***: a vector representation of each training corpus.
- ***Multilingual word representation***: a multilingual word representation obtained by the projection of several pre-trained monolingual embeddings into a unique semantic space (following a linear transformation of each embedding).
- ***ELMo representation***: token-based representation integrating abundant contexts gathered from external resources (Peters et al., 2018).

In this paper, we extend the multilingual graph-based parser proposed by Lim and Poibeau (2017) with the three above representations. Our parser is open source and available at: https://github.com/CoNLL-UD-2018/LATTICE/.

Our parser performed well in the official end-to-end evaluation (73.02 LAS – 4th out of 26 teams, and 78.72 UAS – 2nd out of 26). We obtained very

Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 143–152
Brussels, Belgium, October 31 – November 1, 2018.
https://doi.org/10.18653/v1/K18-2014

good results for French, English and Korean where we were able to extensively exploit the three above features (for these languages, we obtained the best UAS performance on all the treebanks, and among the best LAS performance as well). Unfortunately we were not able to exploit the same strategy for all the languages due to a lack of a GPU and, correspondingly, time for training, and also due a lack of training data for some languages.

The structure of the paper is as follows. We first describe the feature extraction and representation methods (Section 2 and 3) and then present our POS tagger and our parser based on multi-task learning (Section 4). We then give some details on our implementation (Section 5) and we finally provide an analysis of our official results (Section 6).

2 Deep Contextualized Token Representations

The architecture of our parser follows the multilingual LATTICE parser presented in Lim and Poibeau (2017), with the addition of the three feature representations presented in the introduction.

The basic token representations is as follows. Given a sentence of tokens $s=(t_1,t_2,..t_n)$, the i^{th} token t_i can be represented by a vector x_i, which is the result of the concatenation ($\circ$) of a word vector w_i and a character-level vector c_i of t_i:

$$x_i = c_i \circ w_i$$
$$c_i = \text{Char}(t_i; \theta_c)$$
$$w_i = \text{Word}(t_i; \theta_w)$$

When the approach is monolingual, w_i corresponds to the external word embeddings provided by Facebook (Bojanowski et al., 2016). Otherwise we used our own multilingual strategy based on multilingual embeddings (see Section 3.2)

2.1 Character-Level Word Representation

Token t_i can be decomposed as a vector of characters $(ch_1, ch_2,.. ch_m)$ where ch_j is the j^{th} character of t_i. The function Char (that generates the character-level word vector c_i) corresponds to a vector obtained from the hidden state representation h_j of the LSTM, with an initial state h_0 (m is the length of token t_i)[1]:

$$h_j = \text{LSTM}^{(ch)}(h_0, (ch_1,ch_2,..ch_m))_j$$
$$c_i = w_c h_m$$

For LSTM-based character-level representations, previous studies have shown that the last hidden layer h_m represents a summary of all the information based on the input character sequences (Shi et al., 2017). It is then possible to linearly transform this with a parameter w_c so as to get the desired dimensionality. Another representation method involves applying an attention-based linear transformation of the hidden layer matrix H_i, for which attention weights a_i are calculated as follows:

$$a_i = Sofmax(w_{att}\ {H_i}^T)$$
$$c_i = a_i H_i$$

Since we apply the *Softmax* function, making weights sum up to 1 after a linear transformation of H_i with attention parameter w_{att}, the self-attention weight a_i intuitively corresponds to the most informational characters of token t_i for parsing. Finally, by summing up the hidden state H_i of each word according to its attention weights a_i, we obtain our character-level word representation vector for token t_i. Most recently, Dozat et al. (2017) suggested an enhanced character-level representation based on the concatenation of h_m and $a_i H_i$ so as to capture both the summary and context information in one go for parsing. This is an option that could be explored in the future.

After some empirical experiments, we chose bidirectional LSTM encoders rather than a single directional one and then introduced the hidden state H_i into the two-layered Multi-Layer Perceptron (MLP) without bias terms for computing the attention weight a_i:

$$a_i = Sofmax(w_{att2}\ tanh(W_{att1}\ {H_i}^T))$$
$$c_i = a_i H_i$$

For training, we used the charter-level word representations for all the languages except Kazakh and Thai (see Section 5).

2.2 Corpus Representation

Following Lim and Poibeau (2017), we used a one-hot treebank representation strategy to encode

[1] Note that i refers to the i^{th} token in the sentence and that j refers to the j^{th} character of the i^{th} token. Here, we use lowercase italics for vectors and uppercase italics for matrices. So a set of hidden state H_i is a matrix stacked on m characters. In this paper, all the letters w and W denote parameters that the system has to learn.

language-specific features. In other words, each language has its own set of specific lexical features.

For languages with several training corpora (e.g., French-GSD and French-Spoken), our parser computes an additional feature vector taking into account corpus specificities at word level. Following the recent work of Stymne et al. (2018), who proposed a similar approach for treebank representations, we chose to use a 12 dimensional vector for corpus representation. This representation tr_i is concatenated with the token representation x_i:

$$tr_i = \text{Treebank}(t_i; \theta_{\text{tr}})$$
$$x_i = c_i \circ w_i \circ tr_i$$

We used this approach (corpus representation) for 24 corpora, and its effectiveness will be discussed in Section 5.

2.3 Contextualized Representation

ELMo (Embedding from Language Model (Peters et al., 2018)) is a function that provides a representation based on the entire input sentence. ELMo contextualized embedding is a new technique for word representation that has achieved state-of-the-art performance across a wide range of language understanding tasks. This approach is able to capture both subword and contextual information. As stated in the original paper by Peters et al. (2018), the goal is to "learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer".

We trained our language model with bidirectional LSTM using ELMo as an intermediate layer in the bidirectional language model (biLM), and we used ELMo embeddings to improve again the performance of our model.

$$\begin{aligned} R_i &= \{\mathbf{x}_i^{LM}, \overleftrightarrow{\mathbf{h}}_{i,j}^{LM} | = 1, ..., L\} \\ &= \{\mathbf{h}_{i,j}^{LM} | = 0, ..., L\} \end{aligned} \quad (1)$$

$$ELMo_i = E(R_i; \Theta) = \gamma \sum_{j=0}^{L} s_j \mathbf{h}_{i,j}^{LM} \quad (2)$$

In (1), $\mathbf{x}_i^{LM}$ and $\mathrm{h}_{i,0}^{LM}$ are word embedding vectors corresponding to the token layer. $\overleftrightarrow{\mathbf{h}}_{i,j}^{LM}$ is a hidden LSTM vector consisting of a multi-layer and a bidirectional LSTM layer. $\mathbf{h}_{i,j}^{LM}$ is a concatenated vector composed of $\mathbf{x}_i^{LM}$ and $\overleftrightarrow{\mathbf{h}}_{i,j}^{LM}$. We computed our model with all the biLM layers weighted. In (2), s_j is softmax weight that is trainable to normalize multi-layer LSTM layers. γ is the scalar parameter to efficiently train the model. We used a 1024 dimensions ELMo embedding.

3 Multilingual Feature Representations

The supervised, monolingual approach to parsing, based on syntactically annotated corpora, has long been the most common one. However, thanks to recent developments involving powerful word representation methods (a.k.a. word embeddings), it is now possible to develop accurate multilingual lexical models by mapping several monolingual embeddings into a single vector space. This multilingual approach to parsing has yielded encouraging results for both low- (Guo et al., 2015) and high-resource languages (Ammar et al., 2016). In this work, we extend the recent multilingual dependency parsing approach proposed by Lim and Poibeau (2017) that achieved state-of-the-art performance during the last CoNLL shared task by using multilingual embeddings mapped based on bilingual dictionaries.

3.1 Embedding Projection

There are different strategies to produce multilingual word embeddings (Ruder et al., 2018), but a very efficient one consists in simply projecting one word embedding on top of the other to make both representations share the same semantic space (Artetxe et al., 2016). The alternative involves directly generating bilingual word embeddings from bilingual corpora (Gouws et al., 2015; Gouws and Sgaard, 2015), but this requires a large amount of bilingual data aligned at sentence or document level. This kind of resource is not available for most language pairs, especially for under-resourced languages.

We thus chose to train independently monolingual word embeddings and then map these word embeddings one to another. This approach is powerful since monolingual word embeddings generally share a similar structure (especially if they have been trained on similar corpora) and so can be superimposed with little information loss.

To project embeddings, we applied the linear transformation method using bilingual dictionar-

ies proposed by Artetxe et al. (2017). We took the bilingual dictionaries from OPUS[2] and Wikipedia.

The projection method can be described as follows. Let X and Y be the source and target word embedding matrix so that x_i refers to i^{th} word embedding of X and y_j refers to j^{th} word embedding of Y. And let D be a binary matrix where $D_{ij} = 1$, if x_{i} and y_{j} are aligned. Our goal is to find a transformation matrix W such that Wx approximates y. This is done by minimizing the sum of squared errors:

$$\arg\min_{W} \sum_{i=1}^{m} \sum_{j=1}^{n} D_{ij} \|x_i W - y_i\|^2$$

The method is relatively simple since converting a bilingual dictionary into D is quite straightforward. The size of the dictionary used for training is around 250 pairs, and the projected word embedding is around 1.8GB. The dictionaries and the projected word embeddings are publicly available on Github.[3]

3.2 Training with Multilingual Embedding

After having trained multilingual embeddings, we associate them with word representation w_i as follows:

$$w_i = \text{Word}(t_i; \theta_{\text{mw}})$$

We applied the multilingual embedding mostly to train the nine low-resource languages of the 2018 CoNLL evaluation, for which only a handful of annotated sentences were provided.

4 Multi-Task Learning for Tagging and Parsing

In this section, we describe our Part-Of-Speech (POS) tagger and dependency parser using the encoded token representation x_i based on Multi-Task Learning (MTL) (Zhang and Yang, 2017).

4.1 Part-Of-Speech Tagger

As presented in Section 2 and 3, our parser is based on models trained with a combination of features, encoding different contextual information. However, the attention mechanism for the character-level word vector c_i is focusing only on a limited number of features within the token, and the word representation element w_i is thus needed to transform a bidirectional LSTM, as a way to capture the overall context of a sentence. Finally, a token is encoded as a vector g_i:

$$g_i = \text{BiLSTM}^{(\text{pos})}(g_0, (x_1, x_2, ..x_n))_i$$

We transform the token vector g_i to a vector of the desired dimensionality by two-layered MLP with a bias term to classify the best candidate of universal part-of-speech (UPOS):

$$p'_i = W_{pos2}\ leaky\_relu(W_{pos1}\ g_i{}^T) + b_{pos}$$

$$y'_i = \arg\max_j p'_{ij}$$

Finally, we randomly initialize the UPOS embedding as p_i and map the predicted UPOS y'_i as a POS vector:

$$p_i = \text{Pos}(y'_i; \theta_{\text{pos}})$$

4.2 Dependency Parser

To take into account the predicted POS vector on the main target task (i.e. parsing), we concatenate the predicted POS vector p_i with the word representation w_i and then we encode the resulting vector via BiLSTM. This enriches the syntactic representations of the token by back-propagation during training:

$$v_i = \text{BiLSTM}^{(\text{dep})}(v_0, (x_1, x_2, ..x_n))_i$$

Following Dozat and Manning (2016), we used a deep bi-affine classifier to score all the possible head and modifier pairs $Y = (h,m)$. We then selected the best dependency graph based on Eisner's algorithm (Eisner and Satta, 1999). This algorithm tries to find the maximum spanning tree among all the possible graphs:

$$\arg\max_{valid\ Y} \sum_{(h,m)\in Y} Score^{MST}(h, m)$$

With this algorithm, it has been observed that parsing results (for some sentences) can have multiple roots, which is not a desirable feature. We thus followed an empirical method that selects a unique root based on the word order of the sentence, as already proposed by Lim and Poibeau (2017) to ensure tree well-formedness. After the selection

[2] http://opus.nlpl.eu/

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

of the best-scored tree, another bi-affine classifier is applied for the classification of relation labels, based on the predicted tree.

We trained our tagger and parser simultaneously using a single objective function with penalized terms:

$$\begin{aligned} loss = \alpha CrossEntropy(p', p^{(\text{gold})}) \\ + \beta CrossEntropy(arc', arc^{(\text{gold})}) \\ + \gamma CrossEntropy(dep', dep^{(\text{gold})}) \end{aligned}$$

where arc' and dep' refer to the predicted *arc (head)* and *dependency (modifier)* results.

Since UAS directly affects LAS, we assumed that UAS would be crucial for parsing unseen corpora such as *Finnish_PUD*, as well as other corpora from low-resource languages. Therefore, we gave more weight to the parameters predicting arc' than rel' and p', since arc' directly affects UAS. We set $\alpha = 0.1$, $\beta = 0.7$ and $\gamma = 0.2$. Unfortunately, during the testing phase, we did not adjust weight parameters that would have benefited LAS for the 61 big treebanks, and this made our results on big treebanks suffer a bit (7^{th}) compared to those we obtained on Small and PUD treebanks (3^{th}) regarding LAS. This also explains the gap between the UAS and LAS scores in our overall results.

5 Implementation Details

In this section, we provide some details on our implementation for the CoNLL 2018 shared task (Zeman et al., 2018b).

5.1 Training

We have trained both monolingual and multilingual models for parsing. In the first case, we simply used the available Universal Dependency 2.2 corpora for training (Zeman et al., 2018a). In the second case, for the multilingual approach, as both multilingual word embeddings and corresponding training corpora (in the Universal Dependency 2.2 format) were required, we concatenated the corresponding available Universal Dependency 2.2 corpora to artificially create multilingual training corpora.

The number of epochs was set to 200, with one epoch processing the entire training corpus in each language and with a batch size of 32. We then picked the best five performing models to parse the test corpora on TIRA (Potthast et al., 2014). The five models were used as an ensemble run (described in Section 5.2).

Hyperparameters. Each deep learning parser has a number of hyperparameters that can boost the overall performance of the system. In our implementation, most hyperparameter settings were identical to Dozat et al. (2017), except of course those concerning the additional features we have introduced before. We used 100 dimensional character-level word representations with a 200 dimensional MLP, as presented in Section 2, and for corpus representation, we used a 12 dimensional vector. We set the learning-rate to 0.002 with Adam optimization.

Multilingual Embeddings. As described in Section 3, we specifically trained multilingual embedding models for nine low-resource languages. Table 2 gives the list of languages for which we adopted this approach, along with the language used for knowledge transfer. We selected language pairs based on previous studies (Lim and Poibeau, 2017; Lim et al., 2018; Partanen et al., 2018) for *bxr, kk, kmr, sme,* and *hsb*, and the others where chosen based on the public availability of bilingual dictionaries (this explains why we chose to map several languages with English, even when there was no real linguistically motivated reason to do so). Since we could not find any pre-trained embeddings for *pcm_nsc*, we applied a delexicalized parsing approach based on an English monolingual model.

ELMo. We used ELMo weights to train specific models for five languages: Korean, French, English, Japanese and Chinese. ELMo weights were pre-trained using the CoNLL resources provided [4]. We used AllenNLP[5] for training, and used the default hyperparameters. We included ELMo only at the level of the input layer for both training and inference (we set up dropout to 0.5 and used 1024 dimensions for the ELMo embedding layer in our model). All the other hyper-parameters are the same as for our other models (without ELMo).

5.2 Testing

All the tests were done on the TIRA platform provided by the shared task organizers. During the test phase, we applied an ensemble mechanism using five models trained with two different "seeds". The seeds are integers randomly produced by the

[4] http://hdl.handle.net/11234/1-1989
[5] https://github.com/allenai/allennlp

Corpus	UAS	LAS	Rank(UAS)	Rank(LAS)	Baseline(LAS)
Overall (82)	**78.71**	**73.02**	**2**	**4**	65.80
Big treebanks only (61)	85.36	80.97	4	7	74.14
PUD treebanks only (5)	76.81	72.34	3	3	66.63
Small treebanks only (7)	75.67	68.12	2	3	55.01
Low-resource only (9)	37.03	23.39	4	5	17.17

Corpus	Method	UAS(Rank)	LAS(Rank)
af_afribooms		87.42 (7)	83.72 (8)
grc_perseus	*tr*	79.15 (4)	71.63 (8)
grc_proiel	*tr*	79.53 (5)	74.46 (8)
ar_padt		75.96 (8)	71.13 (10)
hy_armtdp	*tr, mu*	**53.56 (1)**	**37.01 (1)**
eu_bdt		85.72 (7)	81.13 (8)
br_keb	*tr, mu*	43.78 (3)	23.65 (5)
bg_btb		92.1 (9)	88.02 (11)
bxr_bdt	*tr, mu*	36.89 (3)	17.16 (4)
ca_ancora		92.83 (6)	89.56 (9)
hr_set		90.18 (8)	84.67 (9)
cs_cac	*tr*	93.43 (2)	91 (2)
cs_fictree	*tr*	**94.78 (1)**	91.62 (3)
cs_pdt	*tr*	92.73 (2)	90.13 (7)
cs_pud	*tr*	89.49 (7)	83.88 (9)
da_ddt		85.36 (8)	80.49 (11)
nl_alpino	*tr*	90.59 (2)	86.13 (5)
nl_lassysmall	*tr*	87.83 (2)	84.02 (4)
en_ewt	*tr, el*	**86.9 (1)**	84.02 (2)
en_gum	*tr, el*	**88.57 (1)**	**85.05 (1)**
en_lines	*tr, el*	**86.01 (1)**	81.44 (2)
en_pud	*tr, el*	**90.83 (1)**	**87.89 (1)**
et_edt		86.25 (7)	82.33 (7)
fo_oft	*tr, mu*	48.64 (9)	25.17 (17)
fi_ftb	*tr*	89.74 (4)	86.54 (6)
fi_pud	*tr*	90.91 (4)	88.12 (6)
fi_tdt	*tr*	88.39 (6)	85.42 (7)
fr_gsd	*tr, el*	**89.5 (1)**	86.17 (3)
fr_sequoia	*tr, el*	**91.81 (1)**	**89.89 (1)**
fr_spoken	*tr, el*	79.47 (2)	73.62 (3)
gl_ctg	*tr*	84.05 (7)	80.63 (10)
gl_treegal	*tr*	78.71 (2)	73.13 (3)
de_gsd		82.09 (8)	76.86 (11)
got_proiel		73 (6)	65.3 (8)
el_gdt		89.29 (8)	86.02 (11)
he_htb		66.54 (9)	62.29 (9)
hi_hdtb		94.44 (8)	90.4 (12)
hu_szeged		80.49 (8)	74.21 (10)
zh_gsd	*tr, el*	71.48 (5)	68.09 (5)
id_gsd		85.03 (3)	77.61 (10)
ga_idt		79.13 (2)	69.1 (4)

Corpus	Method	UAS(Rank)	LAS(Rank)
it_isdt	*tr*	92.41 (6)	89.96 (8)
it_postwita	*tr*	77.52 (6)	72.66 (7)
ja_gsd	*tr, el*	76.4 (6)	74.82 (6)
ja_modern		29.36 (8)	22.71 (8)
kk_ktb	*tr, mu*	39.24 (15)	23.97 (9)
ko_gsd	*tr, el*	88.03 (2)	84.31 (2)
ko_kaist	*tr, el*	**88.92 (1)**	86.32 (4)
kmr_mg	*tr, mu*	38.64 (3)	27.94 (4)
la_ittb	*tr*	87.88 (8)	84.72 (8)
la_perseus	*tr*	75.6 (3)	64.96 (3)
la_proiel	*tr*	73.97 (6)	67.73 (8)
lv_lvtb	*tr*	82.99 (8)	76.91 (11)
pcm_nsc	*tr, mu*	18.15 (21)	11.63 (18)
sme_giella	*tr, mu*	**76.66 (1)**	**69.87 (1)**
no_bokmaal		91.4 (5)	88.43 (11)
no_nynorsk	*tr*	90.78 (8)	87.8 (11)
no_nynorsklia	*tr*	76.17 (2)	68.71 (2)
cu_proiel		77.49 (6)	70.48 (8)
fro_srcmf		91.35 (5)	85.51 (7)
fa_seraji		89.1 (7)	84.8 (10)
pl_lfg	*tr*	95.69 (8)	92.86 (11)
pl_sz	*tr*	92.24 (9)	88.95 (10)
pt_bosque		89.77 (5)	86.84 (7)
ro_rrt		89.8 (8)	84.33 (10)
ru_syntagrus	*tr*	93.1 (4)	91.14 (6)
ru_taiga	*tr*	**79.77 (1)**	74 (2)
sr_set		90.48 (10)	85.74 (11)
sk_snk		86.81 (11)	82.4 (11)
sl_ssj	*tr*	87.18 (10)	84.68 (10)
sl_sst	*tr*	63.64 (3)	57.07 (3)
es_ancora		91.81 (6)	89.25 (7)
sv_lines	*tr*	85.65 (4)	80.88 (6)
sv_pud	*tr*	83.44 (3)	79.1 (4)
sv_talbanken	*tr*	89.02 (4)	85.24 (7)
th_pud	*tr, mu*	0.33 (21)	0.12 (21)
tr_imst		69.06 (7)	60.9 (11)
uk_iu		85.36 (10)	81.33 (9)
hsb_ufal	*tr, mu*	54.01 (2)	43.83 (2)
ur_udtb		87.4 (7)	80.74 (10)
ug_udt		75.11 (6)	62.25 (9)
vi_vtb		49.65 (6)	43.31 (8)

Table 1: Official experiment results for each corpus, where *tr (Treebank)*, *mu (Multilingual)* and *el (ELMo)* in the column Method denote the feature representation methods used (see Section 2 and 3).

Corpus	Projected languages	UAS	LAS
hy_armntdp	Greek	1	1
br_keb	English	3	5
bxr_bdt	Russian	3	4
fo_oft	English	9	17
kk_ktb	Turkish	15	9
kmr_mg	English	3	4
pcm_nsc	-	21	18
sme_giella	Finnish+Russian	1	1
th_giella	English	21	21
hsb_ufal	Polish	2	2

Table 2: Languages trained with multilingual word embeddings and their ranking.

Representation Methods	UAS	LAS
baseline	81.79	78.45
+em	83.39	80.15
+em, tr	83.67	80.64
+em, el	85.47	82.72
+em, tr, el	85.49	82.93

Table 3: Relative contribution of the different representation methods on the overall results.

Python *random* library and are used to initialize the two parameters *W* and *w* (see Section 2). Generally, an ensemble mechanism combines the best performing models obtained from different seeds, so as to ensure robustness and efficiency. In our case, due to a lack of a GPU, different models have been trained simply based on the use of two different seeds. Finally, the five best performing models produced by the two seeds were put together to form the ensemble model. This improved the performances by up to 0.6%, but other improvements could be expected by testing with a larger set of seeds.

5.3 Hardware Resources

The training process for all the language models with the ensemble and ELMo was done using 32 CPUs and 7 GPUs (Geforce 1080Ti) in approximately two weeks. The memory usage of each model depends on the size of external word embeddings (3GB RAM by default plus the amount needed for loading the external embeddings). In the testing phase on the TIRA platform, we submitted our models separately, since testing with a model trained with ELMo takes around three hours. Testing took 46.2 hours for the 82 corpora using 16 CPUs and 16GB RAM.

6 Results

In this section, we discuss the results of our system and the relative contributions of the different features to the global results.

Overall results. The official evaluation results are given in Table1. Our system achieved 73.02 LAS (4th out of 26 teams) and 78.71 UAS (2nd out of 26).

The comparison of our results with those obtained by other teams shows that there is room for improvement regarding preprocessing. For example, our system is 0.86 points below HIT-SCIR (Harbin) for sentence segmentation and 1.03 for tokenization (HIT-SCIR obtained the best overall results). Those two preprocessing tasks (sentence segmentation and tokenization) affect tagging and parsing performance directly. As a result, our parser ranked second on small treebanks (LAS), where most teams used the default segmenter and tokenizer, avoiding the differences on this aspect. In contrast, we achieved 7th on the big treebanks, probably because there is a more significant gap (1.72) here at the tokenization level.

Corpus Representation. Results with corpus representation (corpora marked *tr* in column Method of Table 1) exhibit relatively better performance than those without it, since *tr* makes it possible to capture corpus-oriented features. Results were positive not only for small treebanks (e.g., *cs_fictree* and *ru_taiga*) but also for big treebanks (e.g., *cs_cac* and *ru_syntagrus*). Corpus representation with ELMo shows the best performance for parsing English and French.

Multilinguality. As described in Section 3, we applied the multilingual approach to most of the low-resource languages. The best result is obtained for *hy_armtdp*, while *sme_giella* and *hsb_ufal* also gave satisfactory results. We only applied the delexicalized approach to *pcm_nsc* since we could not find any pre-trained embeddings for this language. We got a relatively poor result for *pcm_nsc*, despite testing different strategies and different feature combinations (we assume that the English model is not fit for it).

Additionally, we found that character-level representation is not always helpful, even in the case of some low-resource languages. When we tested *kk_ktb* (Kazakh) trained with a Turkish corpus, with multilingual word embeddings and character-level representations, the performance dramatically decreased. We suspect this has to do with the writing systems (Arabic versus Latin), but this theory should be further investigated.

sme_giella is another exceptional case since we chose to use a multilingual model trained with three different languages. Although Russian and Finnish do not use the same writing system, applying character and corpus representation improve the results. This is because the size of the training corpus for *sme_giella* is around 900 sentences, which seems to be enough to capture its main characteristics.

Language Model (ELMo). We used ELMo embeddings for five languages: Korean, French, English, Japanese and Chinese (they are marked with *el* in the method column in Table 1). The experiments with ELMo models showed excellent overall performance. All the English corpora, *fr_gsd* and *fr_sequoia* in French, and Korean *ko_kaist* obtained the best UAS. We also obtained the best LAS for English *en_gum* and *en_pud*, and for *fr_sequoia* in French.

Contributions of the Different System Components to the General results. To analyze the

Task	Precision	Recall	F1(Rank)
Event Extraction	58.93	43.12	**49.80 (1)**
Negation Resolution	99.08	41.06	58.06 (12)
Opinion Analysis	63.91	56.88	60.19 (9)

Task	LAS	MLAS	BLEX
Intrinsic Evaluation	84.66 (1)	72.93 (3)	77.62 (1)

Table 4: Official evaluation results on three EPE task (see https://goo.gl/3Fmjke).

effect of the proposed representation methods on parsing, we evaluated four different models with different components. We set our baseline model with a token representation as $x_i = w_i \circ c_i \circ p_i$, where w_i is a randomly initialized word vector, c_i is a character-level word vector and p_i is a POS vector predicted by UDpipe1.1 (note that we did not apply our 2018 POS tagger here, since it is trained jointly with the parser and that affects the overall feature representation). We then initialized the word vector w_i with external word embeddings as provided by the CoNLL shared organizers. We also re-run the experiment by adding treebank and ELMo representations. The results are shown in Table 3 (*em* denotes the use of the external word embedding and *tr* and *el* denotes treebank and ELMo representations respectively.). We observe that each representation improves the overall results. This is especially true regarding LAS when using ELMo (*el*), which means this representation has a positive effect on relation labeling.

Extrinsic Parser Evaluation (EPE 2018). Participants in the CoNLL shared task were invited to also participate in the 2018 Extrinsic Parser Evaluation (EPE) campaign[6] (Fares et al., 2018), as a way to confirm the applicability of the developed methods on practical tasks. Three downstream tasks were proposed this year in the EPE: biomedical event extraction, negation resolution and opinion analysis (and each task was run independently from the others). For this evaluation, participants were only required to send a parsed version of the different corpora received as input back to the organizers using a UD-type format (the organizers then ran the different scripts related to the different tasks and computed the corresponding results). We trained one single English model for the three tasks using the three English corpora provided (*en_lines*, *en_ewt*, *en_gum*) without treebank embeddings (*tr*), since we did not know which corpus embedding would perform better. In addition, we did not apply our ensemble process on TIRA since it would have been too time consuming.

Our results are listed in Table 4. They include an intrinsic evaluation (overall performance of the parser on the different corpora considered as a whole) (Nivre and Fang, 2017) and task-specific evaluations (i.e. results for the three different tasks). In the intrinsic evaluation, we obtained the best LAS among all the participating systems, which confirms the portability of our approach across different domains. As for the task-specific evaluations, we obtained the best result for event extraction, but our parser did not perform so well on negation resolution and opinion analysis. This means that specific developments would be required to properly address the two tasks under consideration, taking semantics into consideration.

7 Conclusion

In this paper, we described the SEx BiST parser (Semantically EXtended Bi-LSTM parser) developed at Lattice for the CoNLL 2018 Shared Task. Our system was an extention of our 2017 parser (Lim and Poibeau, 2017) with three deep contextual representations (multilingual word representation, corpus representations, ELMo representation). It also included a multi-task learning process able to simultaneously handle tagging and parsing. SEx BiST achieved 73.02 LAS (4th over 26 teams), and 78.72 UAS (2nd out of 26), over the 82 test corpora of the evaluation. In the future, we hope to improve our sentence segmenter and our tokenizer since this seems to be the most obvious target for improvements to our system. The generalization of ELMo representation to new languages (beyond what we could do for the 2018 evaluation) should also have a positive effect on the results.

Acknowledgments

KyungTae Lim is supported by the ANR ERA-NET ATLANTIS project. This work is also supported by the LAKME project funded by IDEX PSL (ANR-10-IDEX-0001-02). Lastly, we want to thank the National Science & Technology Information (NTIS) for providing computing resources and the Korean national R&D corpora.

[6] http://epe.nlpl.eu/

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. One parser, many languages. *CoRR* http://arxiv.org/abs/1602.01595.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), Austin, USA*. pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 451–462. aclweb.org/anthology/P17-1042.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2005–2010. https://aclweb.org/anthology/D16-1211.

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.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR* abs/1611.01734. http://arxiv.org/abs/1611.01734.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 20–30. http://www.aclweb.org/anthology/K/K17/K17-3002.pdf.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, pages 457–464.

Murhaf Fares, Stephan Oepen, Lilja vrelid, Jari Björne, and Richard Johansson. 2018. The 2018 Shared Task on Extrinsic Parser Evaluation. On the downstream utility of English Universal Dependency parsers. In *Proceedings of the 22nd Conference on Natural Language Learning*. Brussels, Belgia.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France, volume 37 of *Proceedings of Machine Learning Research*, pages 748–756.

Stephan Gouws and Anders Sgaard. 2015. Simple task-specific bilingual word embeddings. In *HLT-NAACL*. The Association for Computational Linguistics, pages 1386–1390.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of ACL*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *AAAI*. pages 2734–2740.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

KyungTae Lim, Niko Partanen, and Thierry Poibeau. 2018. Multilingual Dependency Parsing for Low-Resource Languages: Case Studies on North Saami and Komi-Zyrian. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan.

KyungTae Lim and Thierry Poibeau. 2017. A system for multilingual dependency parsing based on bidirectional lstm feature representations. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 63–70. http://www.aclweb.org/anthology/K/K17/K17-3006.pdf.

Joakim Nivre and Chiao-Ting Fang. 2017. Universal dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. pages 86–95.

Niko Partanen, KyungTae Lim, Michael Rießler, and Thierry Poibeau. 2018. Dependency parsing of code-switching data with cross-lingual feature representations. In *Proceedings of the Fourth International Workshop on Computatinal Linguistics of Uralic Languages*. pages 1–17.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR* abs/1802.05365. http://arxiv.org/abs/1802.05365.

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.

Sebastian Ruder, Ivan Vuli, and Anders Sgaard. 2018. A survey of cross-lingual word embedding models .

Tianze Shi, Felix G. Wu, Xilun Chen, and Yao Cheng. 2017. Combining global models for parsing universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 31–39. http://www.aclweb.org/anthology/K/K17/K17-3003.pdf.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers)*. Melbourne, Australia.

Dan Zeman et al. 2018a. Universal Dependencies 2.2 CoNLL 2018 shared task development and test data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague, `http://hdl.handle.net/11234/1-2184`. http://hdl.handle.net/11234/1-2184.

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. 2018b. 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.

Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114* .