A Morphology-based Representation Model for LSTM-based Dependency Parsing of Agglutinative Languages

Şaziye Betül Özateş[*], **Arzucan Özgür**[*], **Tunga Güngör**[*], **Balkız Öztürk**[‡]
[*]Department of Computer Engineering
[‡]Department of Linguistics
Boğaziçi University
Bebek, 34342 İstanbul, Turkey
`saziye.bilgin,arzucan.ozgur,gungort,balkiz.ozturk@boun.edu.tr`

Abstract

We propose two word representation models for agglutinative languages that better capture the similarities between words which have similar tasks in sentences. Our models highlight the morphological features in words and embed morphological information into their dense representations. We have tested our models on an LSTM-based dependency parser with character-based word embeddings proposed by Ballesteros et al. (2015). We participated in the CoNLL 2018 Shared Task on multilingual parsing from raw text to universal dependencies as the BOUN team. We show that our morphology-based embedding models improve the parsing performance for most of the agglutinative languages.

1 Introduction

This paper describes our submission to the CoNLL 2018 Shared Task on Universal Dependencies (UD) parsing (Zeman et al., 2018b). We propose morphologically enhanced character-based word embeddings to improve the parsing performance especially for agglutinative languages. We apply our approach to a transition-based dependency parser by Ballesteros et al. (2015) that uses stack Long Short Term Memory structures (LSTMs) to predict the parser state. This parser uses character-level word representation, which has been shown to perform better for languages with rich morphology (Ballesteros et al., 2015; Dozat et al., 2017). From our experiment results performed on UD version 2.2 data sets (Zeman et al., 2018a) we observe that including morphological information to a character-based word embedding model yields a better learning of relationships between words and increases the parsing performance for most of the agglutinative languages with rich morphology.

The rest of the paper is organized as follows: Section 2 provides a brief description of the LSTM-based dependency parser used in this study and introduces our embedding models. Section 3 gives the implementation details of our system and describes the training strategies we apply to different languages. Section 4 discusses our results on the shared task as well as the post-evaluation experiments and Section 5 concludes the paper.

2 Parsing Model

We use the LSTM-based parser by Ballesteros et al. (2015). It is an improved version of a state-of-the-art transition-based dependency parser proposed by Dyer et al. (2015) and uses stack LSTM structures with push and pop operations to learn representations of the parser state. Instead of lookup-based word representations, bidirectional LSTM modules are used to create character-based encodings of words. With this character-based modelling, the authors obtain improvements on the dependency parsing of many morphologically rich languages.

2.1 Character Embeddings of Words

The character-based word embedding model using bi-LSTMs in (Ballesteros et al., 2015) is depicted in Figure 1. The authors compute character-based vector representations of words using bi-LSTMs. Their embedding system reads each word character by character from the beginning to the end and computes an embedding vector of the character sequence, which is denoted as $\vec{w}$ in Figure 1. The system also reads the word character by character from the end to the beginning and the produced embedding is denoted as $\overleftarrow{w}$. These two embedding vectors and the learned representation of the

POS-tag of the word t are concatenated to produce the vector representation of the word.

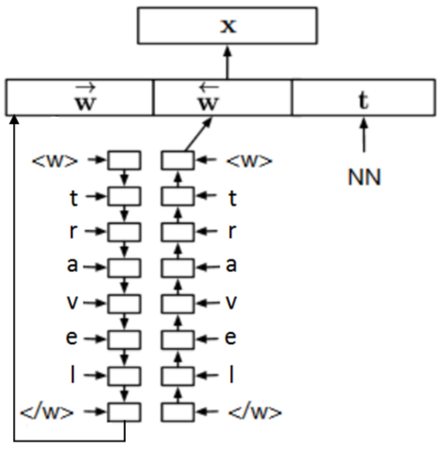

Figure 1: Vector represetntation of the word *travel* with the character-based embedding model in (Ballesteros et al., 2015).

2.2 Morphology-based Character Embeddings

To improve the parsing performance of the LSTM parser with character-based word embeddings mentioned in Section 2.1, we include the morphological information of words to the embedding model. In agglutinative languages like Turkish, a stem usually takes different suffixes and by this way, different meanings are created using a single root-word. Words that share the same suffixes tend to have similar roles in a sentence. Therefore, representing each word using its corresponding lemma and suffixes separately and utilizing the morphological information of words can improve the parsing performance in agglutinative languages.

2.2.1 Lemma-Suffix Model

For agglutinative languages where the stem of a word does not change in different word forms, we created a model that uses lemma and suffix information of words in character-based embeddings. In this model, each word is separated to its lemma and suffixes. Then, the embedding system first reads the lemma of the word character by character from the beginning to the end and computes an embedding vector of the character sequence of the lemma which is denoted as $\vec{r}$. Secondly, the system reads the lemma character by character from the end to the beginning and the produced embedding is denoted as $\overleftarrow{r}$. A similar process is performed for the suffixes of the word and the produced vectors are denoted as $\vec{s}$ and $\overleftarrow{s}$. These four embedding vectors and the vector representation of the POS-tag of the word t are then concatenated to produce the vector representation of the word. Vector representation of an example word using this model is depicted in Figure 2.

2.2.2 Morphological Features Model

The lemma-suffix model is suitable only for agglutinative languages which make use of suffixes to create different word forms. For languages that do not have this type of grammar, we created another model where the specific morphological features of each word are embedded to the dense representations of the words. The reason behind this choice is that words that share the same dependency label usually have similar morphological features.

In this model, the embedding of a word is created character by character as in Section 2.1. Then, the embedding vector of each of its selected morphological features are created by reading the feature value character by character from the beginning to the end. Finally, these embedding vectors and the vector representation of the POS-tag of the word are concatenated to produce the vector representation of the word.

The vector representation of an example word using its morphological features is shown in Figure 3.

3 Implementation

The systems participating in the CoNLL 2018 Shared Task on UD Parsing are expected to parse raw text without any gold-standard pre-processing operations such as tokenization, lemmatization, and morphological analysis. However, the baseline pre-processed versions of the raw text by the UDPipe system (Straka et al., 2016) are available for the participants who want to focus only on the dependency parsing task. We used the automatically annotated version of the corpora provided by UDPipe, since our primary aim is to observe the effect of our embedding models on the dependency parsing of agglutinative languages.

In the implementation of the lemma-suffix embedding model, we did not utilize any morphological analyzer and disambiguator tools to find the lemmas and the suffixes of the words. Instead, for each word in the treebank we extracted its corresponding *lemma* information from the conll-u ver-

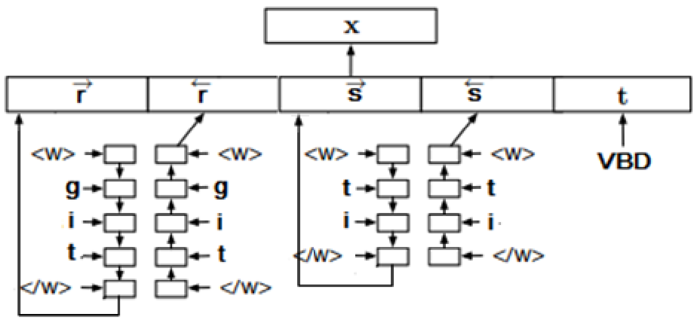

Figure 2: Character-based word embedding of a Turkish word *gitti* (*"it went"* in English) using lemma-suffix embedding model.

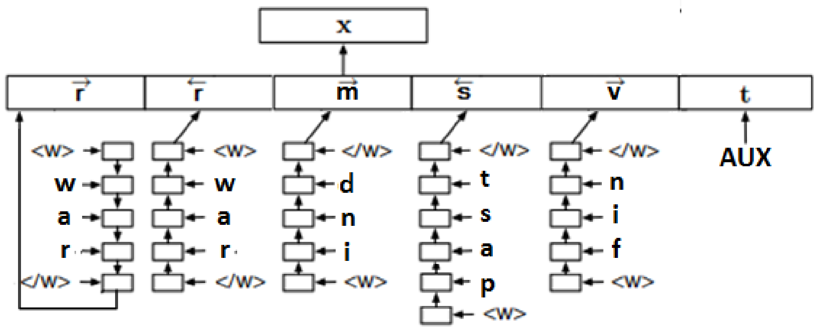

Figure 3: Character-based word embedding of a German word *war* (*"was"* in English) with its morphological features being $Mood = Ind|Number = Sing|Person = 3|Tense = Past|VerbForm = Fin$ using morphological features embedding model. The selected features for German are *Case, Mood, Tense,* and *VerbForm*. Since there is no *Case* feature in the morphological features of *war*, the *Case* feature is represented with an empty string in the word vector of *war*.

sion of the treebank data and subtracted the lemma from the word to find the suffix information. We compared these two approaches on the Turkish-IMST treebank and observed that finding the suffixes by subtracting the lemmas from the words gives the same parsing performance as using a morphological analyzer tool to find the lemma and suffixes of a word. So, we opted not to use a morphological analyzer and disambiguator for the languages with the lemma-suffix embedding model due to the additional costs of these tools.

3.1 Embedding Model Selection for Different Languages

We applied the lemma-suffix model in 2.2.1 to Buryat, Hungarian, Kazakh, Turkish, and Uyghur languages because these languages have agglutinative morphology, take suffixes, and the stem of a word usually does not change in different word forms. We also applied this model to Danish to observe the effect in parsing performance of a language with little inflectional morphology.

For the languages that do not follow this scheme, we applied the morphological features embedding model in 2.2.2. Table 1 shows the morphological features selected for these languages in the shared task. We selected four morphological features from the input conll-u files for most of the languages. For French, Indonesian, and Old French, we used less than four features because there are less than four common morphological features in the conll-u files of these languages.

For Persian, Japanese, Korean, Vietnamese, and Chinese, we used the baseline embedding model due to the lack of representative morphological features in their corresponding conll-u files.

3.1.1 Languages without Training Data

We trained a mixed language parser model with morphological features embedding model for the

Language	**Morphological Features**			
Afrikaans	Aspect	Case	Tense	VerbForm
Ancient Greek	Aspect	Case	Tense	VerbForm
Arabic	Aspect	Case	Mood	VerbForm
Armenian	Aspect	Case	Tense	VerbForm
Basque	Aspect	Case	Tense	VerbForm
Bulgarian	Aspect	Case	Tense	VerbForm
Catalan	AdpType	Mood	Tense	VerbForm
Croatian	Case	Mood	Tense	VerbForm
Czech	Aspect	Case	Tense	VerbForm
Dutch	Degree	Case	Tense	VerbForm
English	Case	Mood	Tense	VerbForm
Estonian	Case	Mood	Tense	VerbForm
French	Mood	Tense	VerbForm	
Finnish	Case	Mood	Tense	VerbForm
Galician	Case	Mood	Tense	VerbForm
German	Case	Mood	Tense	VerbForm
Gothic	Case	Mood	Tense	VerbForm
Greek	Aspect	Case	Tense	VerbForm
Hebrew	HebBinyan	HebSource	Tense	VerbForm
Hindi	Aspect	Case	Tense	VerbForm
Indonesian	PronType	Degree		
Irish	Case	Mood	Tense	VerbForm
Italian	PronType	Mood	Tense	VerbForm
Kurmanji	Case	Mood	Tense	VerbForm
Latin	Case	Mood	Tense	VerbForm
Latvian	Aspect	Case	Tense	VerbForm
North Sami	Case	Mood	Tense	VerbForm
Norwegian	Case	Mood	Tense	VerbForm
Old Church Slavonic	Case	Mood	Tense	VerbForm
Old French	Tense	VerbForm		
Polish	Aspect	Case	Tense	VerbForm
Portuguese	PronType	Mood	Tense	VerbForm
Romanian	Case	Mood	Tense	VerbForm
Russian	Aspect	Case	Tense	VerbForm
Serbian	PronType	Mood	Tense	VerbForm
Slovak	Aspect	Case	Tense	VerbForm
Slovenian	Aspect	Case	Tense	VerbForm
Spanish	Case	Mood	Tense	VerbForm
Swedish	Case	Mood	Tense	VerbForm
Ukrainian	Aspect	Case	Tense	VerbForm
Upper Sorbian	Case	Mood	Tense	VerbForm
Mixed Language	Case	Mood	Tense	VerbForm

Table 1: List of morphological features used for the languages with the morphological features embedding model.

languages with no training data. In the shared task, this model is applied to the *Buryat-KEB, Czech-PUD, English-PUD, Faroese-OFT, Japanese-Modern, Naija-NSC, Swedish-PUD,* and *Thai-PUD* treebanks.

We trained parser models for the *Upper Sorbian-UFAL* and *Galician-TreeGal* treebanks using the morphological features embedding model and for the *Buryat-BDT* treebank using the lemma-suffix embedding model. However, we used the mixed language parser model for these treebanks in the shared task due to some software issues.

3.2 Training Specifications

Our model mostly uses the same hyper-parameter configuration with the original settings of the parser in (Ballesteros et al., 2015) with a few exceptions. We used stochastic gradient descent trainer with a learning rate of 0.13. We replaced the original character-based embedding model with our embedding models. In the lemma-suffix model, the forward word vector and the backward word vector of the lemma of a word both have 50 dimensions. The forward and backward word vectors of the suffix of a word also have 50 dimensions each. In the morphological features model, each of the forward and backward word vectors of a word have 50 dimensions. Each of the four morphological feature vectors have 25 dimensions. If a morphological feature is absent in a word, an embedding vector of an empty string is created for that feature. So, we increased the dimension of the character-based representations to 200 in total.

The original parser is not compatible with UD parsing. We adapted it to be able to take input and produce output in conll-u format. The source code of our modified version of the LSTM-based parser by Ballesteros et al. (2015) can be found at `https://github.com/CoNLL-UD-2018/BOUN`.

A full run over the 82 test sets takes about 3 hours when no pre-trained embeddings are used, and 20 hours when the CoNLL-17 pre-trained word embeddings from (Ginter et al., 2017) are used on the TIRA virtual machine (Potthast et al., 2014). The largest of the test sets needs 4 GB memory without pre-trained word vectors. When the CoNLL-17 pre-trained vectors are used, memory usage can reach to 32 GB depending on the pre-trained vector sizes.

4 Results

This section presents the parsing performance of our parser models on the CoNLL-18 Shared Task as well as the post-evaluation scores of our models.

4.1 Shared Task

Table 2 shows our official LAS, MLAS, and BLEX results in the CoNLL-18 Shared Task. The models that use the CoNLL-17 pre-trained word embeddings from (Ginter et al., 2017) are indicated in *pre-trained vectors* column. We also trained parser models using pre-trained word embeddings for *Czech-PDT, German-GSD, English-EWT, English-GUM, English-LinES, Spanish-AnCora, Indonesian-GSD, Latvian-LVTB, Swedish-LinES, Swedish-Talbanken,* and *Turkish-IMST*. However, we could not run these models with their corresponding embedding files inside the TIRA virtual machine due to some unknown memory and disk issues.

Although the parser we used does not obtain competitive performance when compared with the best performing systems in the shared task, it achieves better performance on the treebanks with no training data when compared to its performance on treebanks with training data. We exclude the parallel UD treebanks from this judgment because one can get better performance on parallel UD treebanks by training the parser using the training data of the treebanks that have the same language with the parallel UD treebanks (e.g., the training data of *English-EWT* for *English-PUD*, *Czech-PDT* for *Czech-PUD* etc.). Due to time-constraints, we did not focus on the parallel UD treebanks and treated them as unknown languages.

4.2 Post-Evaluation

We performed another set of experiments using our models on the test data of UD version 2.2 data sets. The purpose of these experiments is to investigate the effect of our embedding models on parsing performance. Here we used the gold-standard conll-u files instead of the automatically annotated corpora by UDPipe, since our aim in these experiments is to observe the performance difference between our embedding models and the baseline embedding model.

In Table 3, we compare our models with the baseline model proposed in (Ballesteros et al.,

Treebank	**LAS Rank**	**LAS**	**MLAS**	**BLEX**	**Pre-trained vectors**
af-afribooms	23	72.09	58.08	59.42	-
ar-padt	19	66.84	55.68	58.22	ar.vec
bg-btb	21	82.74	72.76	71.14	-
br-keb	10	10.59	0.43	2.21	-
bxr-bdt	19	9.12	1.01	2.46	-
ca-ancora	21	85.03	75.73	76.15	ca.vec
cs-cac	21	83.22	70.50	76.98	cs.vec
cs-fictree	20	82.00	68.75	74.49	cs.vec
cs-pdt	20	83.24	74.04	78.45	-
cs-pud	21	69.60	56.86	63.13	-
cu-proiel	22	60.39	47.99	52.96	cu.vec
da-ddt	21	73.03	63.21	63.69	da.vec
de-gsd	24	56.85	27.51	45.83	-
el-gdt	20	82.11	65.23	68.59	el.vec
en-ewt	22	73.61	64.00	66.11	-
en-gum	22	72.07	60.46	59.93	-
en-lines	22	68.92	59.03	59.74	-
en-pud	23	69.28	56.99	60.04	-
es-ancora	21	83.02	73.95	74.40	-
et-edt	17	75.47	67.74	64.42	et.vec
eu-bdt	19	70.41	57.83	63.36	eu.vec
fa-seraji	19	79.62	73.09	69.84	fa.vec
fi-ftb	20	75.34	65.24	61.69	fi.vec
fi-pud	20	60.07	53.14	48.14	-
fi-tdt	20	75.68	67.56	61.21	fi.vec
fo-oft	19	23.73	0.34	5.81	-
fr-gsd	21	80.07	71.01	72.70	fr.vec
fr-sequoia	21	80.03	70.04	73.03	fr.vec
fr-spoken	25	58.88	45.56	46.02	-
fro-srcmf	22	76.56	68.00	71.17	-
ga-idt	21	55.57	28.92	34.56	-
gl-ctg	18	76.36	62.94	66.00	-
gl-treegal	21	63.43	46.11	48.83	-
got-proiel	21	58.18	44.69	50.63	-
grc-perseus	21	54.57	28.46	35.31	grc.vec
grc-proiel	20	64.77	46.68	52.86	grc.vec
he-htb	21	57.28	43.42	45.94	he.vec
hi-hdtb	21	85.88	67.93	78.21	-
hr-set	22	75.91	56.67	67.43	hr.vec
hsb-ufal	10	29.04	7.18	15.67	-
hu-szeged	22	63.47	50.91	54.57	hu.vec
hy-armtdp	18	21.22	5.66	11.03	-
id-gsd	21	74.01	63.22	62.50	-
it-isdt	23	84.96	75.13	75.21	-
it-postwita	17	67.94	54.15	54.81	it.vec
ja-gsd	21	72.21	58.04	59.78	-
ja-modern	19	18.82	5.02	6.27	-
kk-ktb	21	12.88	2.22	3.95	kk.vec
kmr-mg	21	14.12	2.59	6.65	-
ko-gsd	17	74.99	69.00	62.99	ko.vec
ko-kaist	17	81.45	74.22	68.61	ko.vec
la-ittb	18	76.32	67.78	72.02	la.vec
la-perseus	21	41.94	26.20	28.93	la.vec
la-proiel	21	58.20	45.91	52.11	la.vec
lv-lvtb	20	68.47	54.24	57.43	-
nl-alpino	21	75.94	60.99	63.59	nl.vec
nl-lassysmall	22	74.48	61.40	62.77	nl.vec
no-bokmaal	21	81.47	72.64	73.73	-
no-nynorsk	22	78.48	68.49	69.75	-
no-nynorsklia	21	46.98	35.58	38.54	-
pcm-nsc	19	11.60	3.84	9.60	-
pl-lfg	22	85.11	71.71	75.43	-
pl-sz	22	79.71	61.89	69.76	-
pt-bosque	19	82.62	67.97	72.83	pt.vec
ro-rrt	20	80.18	71.00	71.44	ro.vec
ru-syntagrus	17	84.69	76.43	77.43	ru.vec
ru-taiga	24	45.93	29.00	31.09	-
sk-snk	21	74.37	53.95	59.97	sk.vec
sl-ssj	20	76.78	62.89	68.32	sl.vec
sl-sst	20	44.43	32.07	36.20	sl.vec
sme-giella	20	52.97	42.13	39.10	-
sr-set	23	75.79	62.84	66.68	-
sv-lines	22	72.04	57.75	64.68	-
sv-pud	22	64.55	37.53	48.00	-
sv-talbanken	21	76.93	67.83	68.50	-
th-pud	7	0.70	0.04	0.52	-
tr-imst	22	50.33	40.54	42.00	-
ug-udt	19	55.61	35.98	43.63	ug.vec
uk-iu	20	74.34	56.38	63.46	uk.vec
ur-udtb	21	77.04	50.47	63.40	ur.vec
vi-vtb	20	39.06	25.90	27.61	vi.vec
zh-gsd	22	56.43	46.55	51.20	zh.vec

Table 2: Our official results in the CoNLL-18 Shared Task.

Treebank	**Embedding**	**LAS**		**MLAS**		**BLEX**	
	model	**Baseline**	**Our model**	**Baseline**	**Our model**	**Baseline**	**Our model**
af-afribooms	MF	82.16	**82.80**	73.17	**74.35**	75.48	**76.37**
ar-padt	MF	**78.80**	78.60	**73.59**	73.33	**74.66**	74.39
bg-btb	MF	86.52	**87.41**	80.88	**82.00**	81.33	**82.38**
ca-ancora	MF	**87.21**	87.03	**80.56**	80.23	**81.07**	80.79
cs-cac	MF	87.37	**87.85**	84.12	**84.94**	84.78	**85.49**
cs-fictree	MF	83.49	**86.03**	77.80	**81.64**	78.58	**82.34**
cs-pdt	MF	86.66	**88.47**	83.39	**85.89**	83.91	**86.38**
cu-proiel	MF	**75.73**	75.59	**69.85**	69.62	**72.09**	71.97
da-ddt	LS	77.34	**78.04**	71.45	**71.48**	72.97	**73.33**
de-gsd	MF	77.45	**77.79**	69.79	**70.26**	72.40	**73.24**
el-gdt	MF	83.22	**83.98**	74.83	**76.69**	75.57	**77.53**
en-ewt	MF	**83.88**	83.58	**79.44**	78.95	**79.96**	79.51
en-gum	MF	80.34	**81.46**	73.30	**74.34**	73.87	**75.03**
en-lines	MF	**75.89**	73.83	**71.06**	67.75	**72.53**	69.28
es-ancora	MF	**86.71**	85.55	**80.65**	78.97	**81.17**	79.61
et-edt	MF	80.25	**81.49**	76.59	**78.28**	77.40	**78.98**
eu-bdt	MF	74.07	**74.65**	69.97	**71.21**	71.69	**72.74**
fi-ftb	MF	82.76	**83.88**	77.86	**79.02**	78.54	**79.89**
fi-tdt	MF	80.39	**80.46**	76.33	**76.60**	76.96	**77.31**
fr-gsd	MF	**84.69**	83.77	**78.56**	77.59	**79.13**	78.39
fr-sequoia	MF	**83.16**	82.49	**76.75**	75.96	**77.38**	76.40
fr-spoken	MF	67.99	**68.70**	57.82	**58.19**	58.68	**59.04**
fro-srcmf	MF	**83.01**	82.54	**76.90**	76.44	**77.78**	77.43
ga-idt	MF	61.02	**63.23**	45.21	**47.98**	48.68	**51.90**
gl-ctg	MF	**81.56**	80.70	**70.76**	69.41	**75.38**	74.25
got-proiel	MF	71.88	**74.95**	64.13	**67.99**	66.78	**70.78**
grc-perseus	MF	**61.22**	60.10	**50.75**	49.95	**53.92**	52.79
grc-proiel	MF	79.26	**79.28**	**64.54**	64.12	67.09	**67.16**
he-htb	MF	**80.06**	79.80	**71.56**	70.97	**72.02**	71.52
hi-hdtb	MF	**92.11**	91.51	**87.64**	86.72	**88.38**	87.46
hr-set	MF	80.12	**81.36**	74.75	**76.45**	76.22	**77.90**
hu-szeged	LS	64.00	**68.33**	56.44	**62.55**	59.75	**66.11**
hy-armtdp	MF	**29.60**	28.56	21.65	**25.14**	24.04	**28.56**
it-isdt	MF	88.91	**89.23**	82.77	**83.34**	83.20	**83.79**
it-postwita	MF	**79.19**	79.10	**72.30**	72.26	72.84	**72.91**
kk-ktb	LS	**35.92**	35.34	**25.89**	25.19	30.09	**30.18**
la-ittb	MF	83.86	**85.37**	79.06	**80.93**	80.02	**82.10**
la-perseus	MF	47.48	**51.82**	41.00	**46.56**	44.56	**51.79**
la-proiel	MF	68.81	**70.95**	62.15	**64.66**	64.95	**67.11**
lv-lvtb	MF	73.48	**75.43**	66.19	**68.67**	67.37	**69.66**
nl-alpino	MF	**80.60**	79.11	**72.53**	70.60	**73.25**	71.44
nl-lassysmall	MF	**81.15**	79.19	**74.84**	72.37	**75.52**	73.26
no-bokmaal	MF	**88.53**	88.22	**84.48**	83.87	**84.96**	84.46
no-nynorsk	MF	**86.64**	85.42	**82.00**	80.39	**82.94**	81.21
no-nynorsklia	MF	**66.27**	64.76	**58.40**	56.92	**60.12**	58.55
pl-lfg	MF	92.02	**92.68**	88.93	**89.80**	89.16	**90.01**
pl-sz	MF	85.86	**89.56**	81.34	**86.50**	82.02	**87.17**
pt-bosque	MF	**83.28**	83.20	**75.64**	74.85	**76.95**	76.28
ro-rrt	MF	**81.22**	80.84	**74.26**	74.03	**75.55**	75.36
ru-syntagrus	MF	88.01	**88.14**	84.35	**84.86**	84.72	**85.24**
ru-taiga	MF	48.95	**56.57**	40.82	**50.74**	42.74	**52.33**
sk-snk	MF	78.49	**82.66**	73.94	**79.61**	74.58	**80.46**
sl-ssj	MF	86.23	**88.82**	81.84	**85.33**	82.23	**85.80**
sl-sst	MF	64.47	**65.41**	57.67	**59.38**	59.31	**61.22**
sme-giella	MF	66.21	**71.55**	58.73	**66.87**	61.22	**69.03**
sr-set	MF	**80.71**	80.36	**75.36**	74.84	**76.74**	76.38
sv-lines	MF	76.86	**77.43**	72.98	**73.75**	74.13	**74.72**
sv-talbanken	MF	**83.03**	82.39	**78.36**	77.68	**79.27**	78.58
tr-imst	LS	55.45	**56.74**	49.29	**50.42**	50.45	**51.97**
ug-udt	LS	**58.02**	56.97	**47.47**	45.52	**50.18**	47.86
uk-iu	MF	76.10	**78.87**	70.57	**74.66**	70.77	**74.95**
ur-udtb	MF	**86.07**	86.04	79.43	**79.77**	80.75	**81.02**

Table 3: Comparison of our embedding models with the baseline char-based word embedding model explained in Section 2.1. MF stands for the morphological features embedding model and LS stands for the lemma-suffix embedding model.

Treebank	Number of words	Embedding dimension	LAS	MLAS	BLEX
tr-imst without pre-trained embeddings	-	-	56.74	50.42	51.97
tr-imst with CoNLL-17 ud-word-embeddings	3,633,786	100	59.11	53.02	54.51
tr-imst with Facebook word-embeddings	416,051	300	59.69	53.56	54.98

Table 4: The effect of using pre-trained word embeddings on parsing performance on Turkish-IMST test data set.

2015). Due to time constraints, we trained all models without pre-trained word embeddings.

From the comparative results shown in Table 3, we observe that on the languages that have rich inflectional and derivational processes mostly by adding suffixes to words, our morphological features model outperforms the baseline model in terms of parsing scores. This is the case for the Bulgarian, Croatian, Czech, Basque, Gothic, Latin, Polish, Russian, Slovak, Slovene, North Sami, and Ukrainian languages.

The morphological features model is not suitable for the grammatical structure of Arabic, which has derivational morphology and it also fails to outperform the baseline in Romanic languages like French, Spanish, Catalan, Galician, and Portuguese. The possible reason behind this failure might be the analytic structure of the grammar of these languages. The English, Hebrew, Hindi and Urdu languages are also categorized as mostly analytic languages which do not use inflections and have a low morpheme-per-word ratio. Dutch, Norwegian, and Swedish languages have a very simplified inflectional grammar. So, these languages are not represented well using our morphology-based embedding models. Besides, our model is not the best choice for the languages that have high ratio of morphophonological modifications to the root word like Old Church Slavonic.

The lemma-suffix embedding model is applied to the Danish, Hungarian, Kazakh, Turkish, and Uyghur languages. The best performance is reached in the Hungarian language with more than 4% increase in LAS score. Our model outperforms the baseline in Turkish too. However, the lemma-suffix model fails to reach better performance than the baseline system on the Kazakh and Uyghur treebanks. A possible reason might be that our embedding model increases the complexity of the system unnecessarily for these languages with very little training data. Although Dannish can be considered as an analytic language with a simplified inflectional grammar, the lemma-suffix model outperforms the baseline for this language.

Table 4 shows the parsing scores of the parser with lemma-suffix embedding model on the test data of *Turkish-IMST* treebank version 2.2. We compared the parsing performances when the parser does not use pre-trained word embeddings, when it uses pre-trained embeddings from CoNLL-17 UD word embeddings, and when it uses pre-trained embeddings from word vectors trained on Wikipedia by Facebook (Bojanowski et al., 2017). From the results, we observe that the usage of pre-trained word vectors increases the parsing performance by great extent for Turkish. We also observe that Facebook word vectors outperform the CoNLL-17 UD word vectors, although the number of words in the Facebook vectors data set is much smaller than the number of words in the CoNLL-17 UD word vectors data set.

5 Conclusion

We introduced two morphology-based adaptations of the character-based word embedding model in (Ballesteros et al., 2015) and experimented with these models on the UD version 2.2 data set. The experiment results suggest that our models utilizing morphological information of words increases the parsing performance in agglutinative languages.

Acknowledgments

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant number 117E971 and as a graduate scholarship.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 349–359. https://doi.org/10.18653/v1/D15-1041.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146. http://aclweb.org/anthology/Q17-1010.

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, pages 20–30. https://doi.org/10.18653/v1/K17-3002.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. 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 334–343. https://doi.org/10.3115/v1/P15-1033.

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

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

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

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. 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, pages 1–20.