Subword Contextual Embeddings for Languages with Rich Morphology

Arda Akdemir
University of Tokyo
Tokyo, Japan
aakdemir@hgc.jp

Tetsuo Shibuya
University of Tokyo
Tokyo, Japan
tshibuya@hgc.jp

Tunga Güngör
Boğaziçi University
Istanbul, Turkey
gungort@boun.edu.tr

***Abstract*—Morphological information is important for many sequence labeling tasks in Natural Language Processing (NLP). Yet, existing approaches rely heavily on manual annotations or external software to capture this information. In this study, we propose using subword contextual embeddings for languages with rich morphology. Evaluated on Dependency Parsing (DEP) and Named Entity Recognition (NER) tasks, which are shown to benefit highly from morphological information, subword contextual embeddings consistently outperformed other approaches on all languages tested (Hungarian, Finnish, Czech and Turkish). Our proposed method enables achieving state-of-the-art results with little annotation requirements compared to the previous work. Besides, the novel network architecture we propose, coupled with a Bayesian hyperparameter optimization suite, achieved state-of-the-art results for both tasks for the Turkish language. Finally, we experimented with different multi-task learning architectures to analyze the effect of jointly learning the two tasks.**

***Index Terms*—transfer learning, deep learning, nlp, named entity recognition**

Deep learning based models achieved state-of-the-art results for many sequence labeling tasks in NLP. Representing each input token with a fixed or a trainable vector has shown remarkable progress over previous approaches [1], [2]. However, the token-level approach is not well-suited for languages with rich morphology, where important morpho-syntactic information is retained inside the morphology of the surface form [3]. In addition, out-of-vocabulary and data sparsity pose an important challenge for token-level based systems [4], especially for languages with rich morphology. Fig. 1 shows the coverage of the test vocabulary for increasing sizes of the training set for five languages. Compared to languages with rich morphology, the English language requires the least amount of training data to get equal coverage of the test vocabulary.

In agglutinative languages (e.g., Finnish, Turkish, and Hungarian), words with 3-4 suffixes are quite common, and important information is included in these morphological units rather than within the syntax [3]. For instance, the Turkish word 'geliyordum', which means 'I was coming', contains three suffixes appended to the root 'gel' (to come): '(i)yor' denotes continuous tense, 'du' denotes past tense, and 'm' denotes first person singular. [5] also showed that the vocabulary size drops from 475,975 to 97,734 when only root forms are considered over a corpus of around 10 million words for the Turkish language. This analysis shows that five different words are generated from the same root on average [6], which causes data sparsity. Thus, it is challenging to obtain good word-level vector representations for such languages.

In another group of languages, called inflectional languages (e.g., Czech and Spanish), a single morpheme is used to represent multiple semantic features. For example in Spanish, the -ó in habló, which means 'to speak', simultaneously denotes indicative mode, third person, singular, past tense, and perfective aspect.[1] One of the challenges of entity recognition for inflectional languages like Czech is that the surface form of the words change depending on the case. 'I go to Tokyo' is translated to Czech as 'Jdu do **Tokia**' (dative), and 'I live in Tokyo' is translated as 'Bydlím v **Tokiu**' (locative). The word 'Tokio' becomes 'Tokia' and 'Tokiu' for dative and locative cases, respectively. This change in the surface forms makes it difficult for the token-based models to capture good representations for proper nouns.

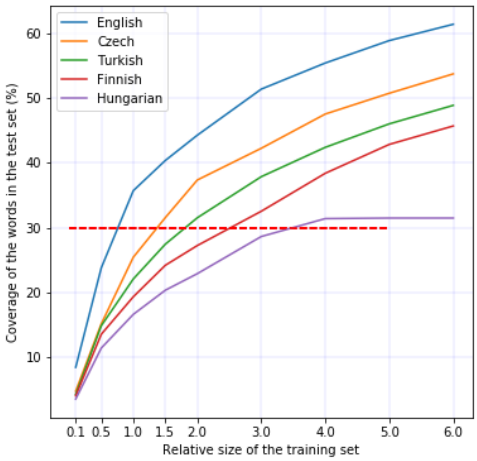

Fig. 1. The coverage of the test set vocabulary for increasing sizes of the training sets (relative to the size of the test sets). The Hungarian language requires around four times more training data to have equal coverage with the English language.

[1]https://glossary.sil.org/term/fusional-language

Linguistically motivated attempts focus on capturing the morphological information by dividing the tokens into subword lexical units and use the vector representations of these units to represent a token [4]. Early approaches mostly use character n-gram features [7] or morphological analyses [3], [8]. Recent works make use of character n-gram embeddings to capture the subword information [9], [10].

The importance of using morphological features and morpheme embeddings for morphologically rich languages has been shown in various studies. Specifically [3], [11], and [6] showed significant performance improvements for named entity recognition (NER) by using either hand-crafted morphological features or by using morphological embeddings. [12] denoted lemmatization as a necessary step for Czech NER systems, and compared different lemmatization approaches. [8] obtained significant gains for dependency parsig (DEP) by using a lexeme-based rather than a token-based approach, where the lexical units are obtained using a morphological analyzer. Yet, all models require either a manually annotated dataset for morphological analysis or an external analyzer for annotation.

Multi-task Learning (MTL) and Language Modeling (LM) have both seen remarkable breakthroughs in recent years. MTL is shown to boost the performance of high-level tasks by leveraging the information obtained in low level tasks [1], [13] and preventing deep learning models from overfitting a single task domain. Language Models trained on huge unlabeled datasets such as ELMo [14] and BERT [15] are successfully applied to many downstream NLP tasks.

The above findings and challenges motivated us to analyze the effect of using subword contextual embeddings for languages with rich morphology. We claim that subword contextual embeddings improve the performance over other approaches for these languages. We use multilingual BERT [15] (mBERT) to obtain the subword contextual embeddings, and compare the performance with using word-level Word2Vec embeddings [2], and subword-level non-contextual FastText embeddings [9]. To test our claim, we picked two NLP tasks for which morphological information is shown to be critical [3], [8]: Dependency Parsing and Named Entity Recognition. Empirical results on four morphologically rich languages (Hungarian, Finnish, Czech, and Turkish) verified our claim for both tasks. mBERT embeddings significantly outperformed the other approaches on all languages tested. Besides, we incorporated a Bayesian hyperparameter optimization to obtain state-of-the-art results for both tasks on commonly used benchmarks for the Turkish language. Finally, we experimented with different multi-task learning architectures to see if the information obtained from each task can be utilized to improve the overall performance. Our main contributions can be listed as follows:

- We analyze using subword contextual representations on four morphologically rich languages for two sequence labeling tasks.
- We outperfom the previous state-of-the-art models on both tasks for the Turkish language by 1) using subword contextual embeddings and 2) implementing a novel neural-network architecture.
- We experiment with different multi-task learning architectures and analyze the effect of information sharing for NER and DEP tasks.

Our work extends the previous work on using subword contextual embeddings for sequence labeling [16], [17] and using subword contextual embeddings with multi-task learning [18]. Recently, [19] also used BERT and Flair [20] contextual embeddings to improve the performance of their neural network baselines for the NER and DEP tasks. However, they do not give a comparison with other embedding approaches and their work is limited to the Czech language. [21] evaluated the performance of mBERT for NER and DEP tasks for 39 languages. Our work differs from them in that we give a comparison with other embedding approaches using the same neural network architecture (embedding-level comparison), whereas they compare mBERT results with the best-published results (model-level comparison). [16] proposed a similar dependency parser which incorporates word-level ELMo embeddings [14] and obtained the highest LAS score (overall and for the Turkish language) on the 'CoNLL 2018 Shared Task'. Our final model outperformed their approach for the Turkish language by using minimal features.

I. Methodology

A. Contextual Subword Embeddings

Representing words with pretrained and fixed vectors learned over huge unlabeled datasets [2] has been a significant breakthrough in NLP research. However, word-level vector representations are not capable of capturing the sub-word level information, which is important for many sequence labeling tasks such as NER, part-of-speech (POS) tagging, and DEP [4]. Another main drawback of these approaches, and of non-contextual vector representations in general, is that they are independent of the context once they are learned. For example, the word 'bass' in 'I like eating bass' and 'I play bass guitar' would be represented with the same vector even though they refer to different senses. Recent works on contextual representations overcome this problem by using a Long Short Term Memory (LSTM) [22] or a Transformer model [23] to condition the output for a word to its surrounding context.

In this study, we use multilingual BERT [15], which is a sub-word level pretrained Transformer-based language model for multiple languages. We claim that the information retained in the morphological units of a word can be captured through the contextual subword representations generated by BERT.

B. Input Layer

The input layer is the concatenation of three vectors. For each word w_j in an input sentence, the output is represented as $o_j = v_j^{word} \oplus v_j^{cas} \oplus v_j^{pos}$, where v_j^{word}, v_j^{cas}, and v_j^{pos} correspond to the word embedding, casing embedding, and POS tag embedding, respectively. For Word2vec embeddings, v_j^{word} is the vector corresponding to that word if the word is included in the training vocabulary or is the vector used

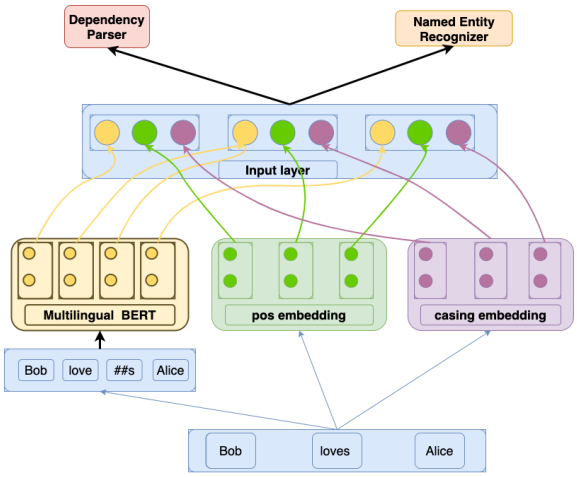

Fig. 2. Overview of the input for the task-specific components. These components are trained jointly in the multi-task settings.

to represent all unknown words. For FastText embeddings, v_j^{word} is the combination of the vectors of its n-grams if the word is not included in the training vocabulary. To obtain v_j^{word} for the mBERT case, we first tokenize the word into its BERT subtokens. Then for each subtoken, we get the mean of hidden vectors of the final four layers of BERT following the suggestions of [15]. Finally, we take the average over all subtokens to get the final representation.

Lower casing is a common preprocessing step when using word embeddings to reduce the vocabulary size. To retain the casing information, which is shown to be useful for the NER task [3], [17], we use casing embeddings with five categories. An example to each category is as follows : 'Title', 'ALL-CAPS', 'lower' , 'contains'apostrophe' and '1234' (numeric). For POS embeddings, we used the XPOS, language-specific POS tags defined in the Universal Dependency format [24]. The input layer is trained jointly on the multi-task settings.

C. Dependency Parser

The dependency parser we used in this work is heavily influenced by the graph-based parsers proposed by [25] and [26]. It consists of two modules to generate scores for arcs between words (denoting dependencies), and labels for each arc (denoting dependency type), separately. We first followed [25] and incorporated a Highway-LSTM (HLSTM) architecture [27] as the initial task-specific layer. HLSTM learns adjustable parameters for controlling the information flow between LSTM layers. Given a sequence of n vectors $\{o_t : 1 \leq t \leq n\}$, we obtain

$$h_t = HLSTM^{dep}(o_1, ..., o_n; t)$$

where n is the number of tokens and h_t is the hidden representation for o_t. [26] make a key observation that the vector representation of a word should be different when it is considered as a head or a dependent. Following this observation, these representations are fed into two Fully Connected

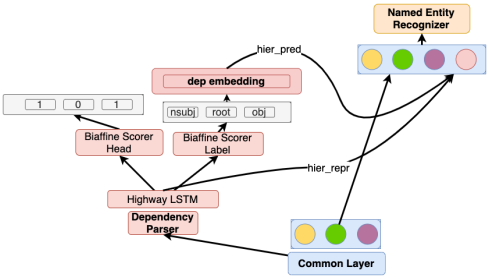

Fig. 3. The detailed architecture of the hierarchical multitask setting when DEP is modeled to be the low-level task.

layers (FC_{head}^{edge} and FC_{dep}^{edge}) to generate representations for each word as a head and as a dependent. Unlike the reference parsers, which use the concatenation of six different embeddings, we only make use of two embeddings in addition to the word embedding. Our parser with mBERT embeddings outperformed both parsers, when trained and tested on the same settings.

D. Named Entity Recognizer

The named entity recognizer consists of three layers: LSTM layer, Fully Connected layer, and Conditional Random Fields (CRF) layer. This combination of LSTM and CRF is commonly used for the NER task [10], [28]. We extend the previous work on BiLSTM-CRF based architectures by replacing the BiLSTM layer with the Highway-LSTM architecture [27]. The architecture of the Highway-LSTM is identical with the DEP component.

E. Multi-task Learning Framework

After verifying our initial claim on using subword contextual embeddings for languages with rich morphology, and achieving state-of-the-art results for both tasks for the Turkish language, we also analyzed the effect of learning the two tasks in different multi-task learning settings.

For the MTL framework, we considered three different architectures where the main difference is how the task-specific components communicate. The first model, $flat$, is similar to the Multi Task Deep Neural Networks (MT-DNN) framework [18], which uses *hard-sharing*. In *hard-sharing*, task-specific components share a common low-level layer, and the arcs in Fig. 3 from the parser to the NER component (hier_repr and hier_pred) are null. For the second and third models, we followed [13], and implemented two hierarchical models. In the second setting, which we call $hier\_pred$ as an abbreviation for 'hierarchical and prediction', the higher-level component receives the embedding of the prediction made by the low-level task component. First, the DEP component gets the input sentence and makes a dependency label prediction for each word. Next, we use an embedding layer to generate the vector representation of the label type. To get the label representation, we first use a simplistic assumption and pick

the highest scoring label of the highest-scoring arc for each word w_i.

$$
\begin{aligned}
head_i &= \underset{j \in \{1,\ldots,n\}}{\text{argmax}} \quad s^{arc}_{i,j} \\
label\_max_i &= \underset{l \in L}{\text{argmax}} \quad p_{i,head_i} \\
v^{dep}_i &= \text{DEP\_EMBED}(label\_max_i) \\
o_i &= v^{word}_i \oplus v^{cas}_i \oplus v^{pos}_i \\
v_i &= o_i \oplus v^{dep}_i
\end{aligned}
$$

where $s^{arc}_{i,j}$ is the score of the arc from w_i to w_j, L is the label set for the dependency task, $p_{i,head_i}$ is the vector containing the score for each label for an arc from w_i to w_{head_i}, $label\_max_i$ is the highest scoring label for this arc, DEP_EMBED is the function that maps a dependency label to a fixed-size vector, and v_i is the input vector to the NER component. We refer to this method of inputting the prediction embedding as 'hard'. Alternatively, we also used a weighted average of the embeddings, which we refer to as 'soft', for each dependency label type based on the scores generated for each of them, also employed by [13]. In this setting v^{dep}_i is defined as

$$
v^{dep}_i = \sum_{l \in L} \text{DEP\_EMBED}(l) \frac{exp(p_{i,head_i}[l])}{\sum_{l \in L} exp(p_{i,head_i}[l])}
$$

For the third setting, which we call 'hier_repr' to denote 'hierarchical representation', we consider directly concatenating the hidden layer output generated for the DEP component (arrow denoted as 'hier_repr' on Fig. 3).

$$
\begin{aligned}
h^{dep}_i &= HLSTM^{dep}(o_1, ..., o_n; i) \\
o_i &= v^{word}_i \oplus v^{cas}_i \oplus v^{pos}_i \\
v_i &= o_i \oplus h^{dep}_i
\end{aligned}
$$

where h^{dep}_i is the final hidden layer output of the HLSTM of the DEP component for w_i (see Appendix A for details regarding the implementation of each task-specific coomponent).

II. Experimental Settings

For all the experiments, we used Tesla V100 GPU's with a single thread. The multitask model with the highest number of parameters trains at a speed of around 200 sentences/second. During the inference mode, the model can output predictions with 600 sentences/second (see Appendix C1 for more details about the experiments).

A. Datasets

For each language, we used the largest datasets provided in 'CoNLL 2018 Universal Dependencies Shared Task' [24] for the DEP task. The datasets are annotated in CoNLL-U format and contain XPOS tags in addition to the head and dependency label for each token. For evaluation, we report results using the two most common scoring metrics for the DEP task: Labeled Attachment Score (LAS) and Unlabeled Attachment Score (UAS). LAS considers a prediction correct only if both the head index and the dependency label are predicted correctly. For UAS the latter constraint is removed.

The Turkish NER dataset is a commonly used benchmark [29], which contains named entity labels for Location (LOC), Organization (ORG), and Person (PER). The Hungarian NER dataset [30] also contains MISC type, in addition to the labels inside the Turkish dataset. CNEC 2.0 [31] corpus was used for the Czech language, which contains two-level annotations for each named entity. For all experiments, we used the seven supertypes as the named entity labels. Finally, for the Finnish language, we used a dataset compiled from news articles [32]. The dataset is annotated for ten entity types. For all languages, the annotation is done using the IOB-2 tagging scheme, where the first token of each entity has the prefix 'B-' and the remaining tokens are prefixed with an 'I-'. To evaluate the performance of the NER component, we use the micro F-1 score over all entities. All details regarding each dataset is given in Appendix C1.

B. Training Details

For all the reported results of our models, we used a default batch-size of 500 words. We define an epoch as 100 steps to have a roughly equal evaluation interval with the StanfordNLP dependency parser [26]. We evaluated the performance on the validation datasets, and used early stopping with patience, and stopped the training when we could not observe any improvements on the validation splits (see Appendix C1 and C1 for more details about the training).

1) Pretraining word embeddings: Publicly available pretrained models for Word2Vec and FastText use up to 300-dim vectors to represent each word, whereas mBERT uses 768-dim vectors. Increasing the dimensions may provide an advantage to capture more subtle information about a word. To have a fair comparison between the proposed method of using the subword contextual embeddings and the other embedding methods, we pretrained both the FastText and the Word2vec embeddings to have matching 768-dim with the mBERT vectors. For all languages, we used the latest Wikipedia dumps [2] as the training dataset, and applied the same preprocessing scripts to prepare the datasets. We share all the necessary code to obtain and process the datasets, and pretrain both embeddings. As the baseline for word embeddings, we randomly initialized 768-dim vectors for each word in the training set, and learned the vector representations during the training process. For token-based methods, a special 'UNK' token is used to represent all the words that do not occur inside the training sets.

2) Hyperparameter optimization: In deep-learning based MTL settings, there are various hyperparameters regarding the architecture and the training regime. An eager attempt to exhaustively search over the entire hyperparameter space is almost always infeasible. To find a good hyperparameter configuration, we used a Bayesian optimization based hyperparameter optimizer [33]. We ran the optimizer for 50 trials

[2]https://dumps.wikimedia.org/backup-index.html

TABLE I
RESULTS FOR USING DIFFERENT EMBEDDING TYPES IN THE SINGLE-TASK LEARNING (STL) (TOP) AND MULTI-TASK LEARNING (MTL) (BOTTOM)

Model	Language	Embedding type	NER Results			DEP Results	
			Precision	**Recall**	**F-1**	**LAS**	**UAS**
STL	Czech	Baseline	56.40	53.57	54.95	72.43	78.91
		Word2vec	63.46	48.65	55.07	78.09	82.72
		FastText	62.11	51.40	56.25	77.96	84.10
		mBERT	**77.30**	**75.63**	**76.46**	**88.28**	**91.17**
	Hungarian	Baseline	86.44	78.33	82.19	59.51	70.85
		Word2vec	90.08	88.96	89.52	59.76	70.03
		FastText	90.24	84.10	87.06	61.85	72.66
		mBERT	**95.16**	**95.56**	**95.36**	**69.19**	**75.34**
	Finnish	Baseline	77.04	63.55	69.65	61.25	69.19
		Word2vec	74.78	60.92	67.14	64.61	72.90
		FastText	76.68	61.44	68.22	72.51	78.74
		mBERT	**82.92**	**84.95**	**83.92**	**83.78**	**86.65**
	Turkish	Baseline	73.75	80.28	76.88	55.91	64.57
		Word2vec	**91.04**	78.10	84.07	57.41	65.94
		FastText	89.17	80.93	84.85	60.14	67.54
		mBERT	90.09	**90.36**	**90.22**	**64.28**	**70.54**
MTL	Czech	Baseline	66.63	52.98	59.02	76.32	82.20
		Word2vec	63.36	49.73	55.72	75.24	80.98
		FastText	64.09	52.29	57.59	75.31	80.43
		mBERT	**76.23**	**76.12**	**76.18**	**84.98**	**89.05**
	Hungarian	Baseline	87.48	78.12	82.54	57.91	68.36
		Word2vec	93.48	83.68	88.31	58.48	69.84
		FastText	93.00	86.67	89.72	62.21	71.61
		mBERT	**95.00**	**94.93**	**94.96**	**70.46**	**76.41**
	Finnish	Baseline	69.94	59.49	64.29	61.26	68.78
		Word2vec	73.88	59.38	65.84	65.07	72.94
		FastText	72.01	60.16	65.55	64.39	71.30
		mBERT	**81.64**	**83.71**	**82.66**	**83.78**	**86.67**
	Turkish	Baseline	91.06	78.68	84.42	57.16	65.76
		Word2vec	90.80	79.04	84.51	58.47	66.32
		FastText	**92.14**	79.95	85.61	59.92	67.68
		mBERT	88.79	**90.46**	**89.62**	**63.85**	**69.97**

for each task-specific component separately. The objective function of the optimizer to minimize is the negative of the evaluation metric for each task. For each such configuration, training is done until at most 40 epochs, and the configuration with the highest F-1 score is used. For all training settings, we used learning rate patience and early stopping (see Appendix C1 for all details about the hyperparameters).

III. RESULTS

A. Comparison of word embedding types

We started our experiments by verifying our initial claim on using subword contextual embeddings for languages with rich morphology. We trained and tested the task-specific components separately for all languages (single-task learning). Table I shows the results of using different word embeddings as input to the neural networks. 'Baseline' refers to using randomly initialized trainable vectors for each word inside the training sets.

mBERT embeddings outperformed other approaches significantly, for both tasks and for all languages. Using Word2Vec and FastText word embeddings brought improvement over using randomly initialized vectors except for the NER task for the Finnish language. Moreover, subword-based FastText embeddings outperformed the token-level Word2vec embeddings on all configurations except for the NER task for the Hungarian.

These results verified our initial claim on using subword-based embeddings for languages with rich morphology. We observed that the performance improvement is more significant for the Czech and Finnish languages, especially for the NER task (20.21 and 15.70 absolute F-1 score increase, respectively).

B. Factors that make mBERT standout

To better understand where the improvements for mBERT come from, we analyzed the results on the NER test sets in the single-task learning setting. We first examined the test sentences for which other approaches failed, and mBERT succeeded, for all languages. Upper part of Table II shows the results of this analysis. The last column (Overall) gives the statistics for all the test set sentences, for comparison.

Unknown entities. These are the entities that appear in the test sets, and do not appear in the training sets. We observed that these sentences, except for the Word2Vec-Finnish combination, have a higher unknown entity frequency than the whole test sets of each language. This observation supports our initial claim that unknown words pose a challenge to the compared models, and mBERT is a better alternative to cope with them.

Rare entities. We define rare words as words that only occur 1-3 times inside the training datasets of each language. Similar to the unknown words, in most cases (all languages except Finnish, and Baseline-Turkish combination) these sentences have higher rare word frequencies. These results support our claim that mBERT is better at handling rare words compared to the other approaches.

The results above also showed that the Finnish and Czech test sets have much higher unknown-rare entity frequencies in comparison to the Hungarian and Turkish languages. This may explain why the best results for the NER task are significantly lower for these languages (76.46 and 83.92 for the Czech and Finnish languages, and 95.36 and 90.22 for the Hungarian and Turkish languages, respectively). Besides, this may also explain why the performance improvement is more significant for these languages.

Long entities. We observed that for all languages and all the compared embedding methods, these sentences contained longer entities. For languages with rich morphology, word length is a good estimator for suffixation. This analysis supports that some of the improvements attained by mBERT over others is because of its syntactic capabilities [34].

Long sentences. Except for the Czech language, the results showed that, for all models, these sentences are longer than the average. For the Turkish language, the average lengths of these sentences are roughly 25% higher than the overall value. This increase suggests that the contextuality of mBERT helps the model to perform better on instances that contain more contextual information.

Entities with suffixation. In the Turkish language, suffixes to a proper noun are separated with the ' sign e.g., -de suffix denoting locative case in 'Türkiye'de' which means 'in Turkey'. Using this fact, we further analyzed the performance over the Turkish entities containing at least one suffix (has ' sign inside the token). We observed that the frequency of entities with suffixes are much higher among the examples that

TABLE II
ANALYSIS OF THE RESULTS ON THE NER TEST SETS IN THE SINGLE-TASK LEARNING SETTING, FOR ALL LANGUAGES

	Czech				Hungarian				Finnish				Turkish			
	Baseline	**Word2Vec**	**FastText**	**Overall**	**Baseline**	**Word2Vec**	**FastText**	**Overall**	**Baseline**	**Word2Vec**	**FastText**	**Overall**	**Baseline**	**Word2Vec**	**FastText**	**Overall**
mBERT succeded, and others failed.																
Number of sentences	154	164	154	889	182	95	136	933	746	813	760	3506	385	274	254	2729
Number of entities	659	691	659	2473	629	326	504	1161	1544	1611	1568	3174	638	652	611	1667
Unknown entity frequency	0.551	0.553	0.568	0.550	0.501	0.525	0.530	0.411	0.552	0.549	0.561	0.552	0.379	0.445	0.453	0.369
Rare entity frequency	0.739	0.738	0.751	0.736	0.628	0.653	0.645	0.597	0.683	0.689	0.689	0.701	0.514	0.557	0.563	0.525
Average entity length	5.62	5.64	5.76	5.44	6.90	6.87	6.96	6.81	7.89	7.91	7.93	7.57	7.16	7.31	7.21	6.91
Average sentence length	22.1	22.3	22.1	22.5	28.5	27.6	28.6	24.1	14.6	14.6	14.7	13.4	17.0	17.8	18.2	13.2
mBERT failed, and others succeded.																
Number of sentences	26	27	25	889	10	13	14	933	144	115	139	3506	79	85	95	2729
Number of entities	77	80	69	2473	26	50	50	1161	284	217	254	3174	168	175	240	1667
Unknown entity frequency	0.519	0.412	0.507	0.550	0.346	0.4	0.34	0.411	0.352	0.341	0.319	0.552	0.304	0.309	0.312	0.369
Rare entity frequency	0.636	0.625	0.667	0.736	0.423	0.580	0.500	0.597	0.433	0.424	0.409	0.701	0.423	0.429	0.450	0.525
Average entity length	5.07	5.28	5.20	5.44	5.59	6.66	7.26	6.81	7.73	7.85	7.69	7.57	6.97	6.56	6.69	6.91
Average sentence length	24.4	24.9	23.8	22.5	29.5	31.4	27.6	24.1	13.9	13.2	13.2	13.4	17.6	16.5	18.6	13.2

other embedding approaches failed. Only 30.8% of all named entities contain a suffix in the Turkish NER dataset. This value increases to 56.7% when we only consider the entities that all other embedding-based models failed to detect correctly. This significant increase in frequency suggests that these models struggle with languages with high suffixation.

We repeated the same analysis for the opposite cases (sentences that mBERT failed, and other methods succeeded.). This analysis is necessary to validate that the findings mentioned above are not just characteristics of any 'difficult' sentences. Lower part of Table II shows that similar observations can not be made on these sentences. These sentences have lower unknown-rare entity frequencies. Besides, the entity and sentence lengths do not follow a clear trend.

C. Effects of multi-task learning

Next, we trained the task-specific components jointly in a multi-task learning setting. We used the *hard-sharing* based 'flat' model as the multi-task learning architecture, where the input layer is shared between task-specific components. For all experiments, the architecture of the joint model is kept the same as those of the individual models. Bottom part of Table I shows the results for the multi-task learning for all languages and both tasks. Similar to the single-task learning setting, mBERT based multi-task learning model significantly outperformed other models. Yet, we observed that the proposed mBERT based model did not benefit from learning these tasks jointly except for the DEP task for the Hungarian language (+1.27 absolute LAS score). Overall, the benefits of *hard-sharing* based multi-task learning were unclear for these tasks.

D. Effects of hyperparameter optimization

The above findings and observations motivated us to apply the Bayesian hyperparameter optimization, explained in Section II-B2, to get the final performance on both tasks for the Turkish language. Table III shows the results for each task, along with the previous state-of-the-art results on the same datasets. On top, we give the results obtained by the proposed mBERT based models. These are followed by the previous results on the DEP and the NER datasets for the Turkish language, respectively. All results given in Table III are obtained on the same splits for both datasets. The results show that the proposed mBERT based model outperformed

TABLE III
RESULTS FOR BOTH TASKS FOR THE TURKISH LANGUAGE AFTER HYPERPARAMETER OPTIMIZATION

Model	NER F-1	DEP LAS
mBERT	**93.82**	67.88
mBERT + MTL (flat)	92.80	68.52
mBERT + MTL (hier_repr)	93.72	68.07
mBERT + MTL (hier_pred)	93.17	**68.87**
Akdemir et al. [35]	89.21	50.01
Dozat et al. [26]	-	64.86
Che et al. [16]	-	66.44
Yeniterzi et al. [11]	88.94	-
Demir et al. [6]	91.96	-
Seker et al. [36]	91.94	-
Gungor et al. [3]	93.59	-
Gunes et al. [37]	93.69	-

state-of-the-art models on both tasks. Finally, we analyzed the effect of using different multi-task learning architectures explained in Section I-E, in this same setting. All multi-task learning architectures brought additional improvement for the DEP task. Yet, the performance on the NER task degraded when we incorporated multi-task learning. Besides, except for 'hier_repr' for the DEP task, hierarchical settings outperformed the 'flat' model, which is in line with the observations made by Hashimoto et al. [13]. We obtained +2.43% absolute LAS score for the DEP task and +0.13% absolute F-1 score for the NER task over the state-of-the-art results. The models achieved this by using minimal features (pos tags and casing features) compared to the competitive models. In addition, compared to the multi-task learner in the same setting [35], we obtained significant improvements for both tasks (+18.86% absolute LAS for the DEP task and +4.61% absolute F-1 for the NER task).

IV. CONCLUSION

In this study, we claimed that using subword contextual embeddings improves performance for languages with rich morphology. We implemented two neural network models following the state-of-the-art approaches for the DEP and the NER tasks. The empirical results showed that using subword contextual embeddings outperforms other approaches. The error analysis revealed that improvements for mBERT came from its ability to handle rare-unknown entities, and long

words and sentences, for the NER task. We also analyzed the effect of incorporating multi-task learning. We observed that the hierarchical settings consistently outperformed the 'flat' model. Finally, after applying hyperparameter optimization, the proposed model outperformed the state-of-the-art models for both DEP and NER for the Turkish language.

Further gains may be possible by incorporating character-level embeddings, and through extending the multi-task framework to include more tasks. To promote future research, we make all the source-code publicly available. [3]

V. Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 17H01693, 20K21827, and CREST Grant Number JPMJCR1402.

References

[1] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[3] O. Güngör, S. Üsküdarlı, and T. Güngör, "Improving named entity recognition by jointly learning to disambiguate morphological tags," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 2082–2092.

[4] T. Luong, R. Socher, and C. Manning, "Better word representations with recursive neural networks for morphology," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, 2013, pp. 104–113.

[5] D. Z. Hakkani-Tür, K. Oflazer, and G. Tür, "Statistical morphological disambiguation for agglutinative languages," *Computers and the Humanities*, vol. 36, no. 4, pp. 381–410, 2002.

[6] H. Demir and A. Özgür, "Improving named entity recognition for morphologically rich languages using word embeddings." in *ICMLA*, 2014, pp. 117–122.

[7] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[8] G. Eryiğit, J. Nivre, and K. Oflazer, "Dependency parsing of turkish," *Computational Linguistics*, vol. 34, no. 3, pp. 357–389, 2008.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[10] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 260–270.

[11] R. Yeniterzi, "Exploiting morphology in turkish named entity recognition system," in *Proceedings of the ACL 2011 Student Session*. Association for Computational Linguistics, 2011, pp. 105–110.

[12] M. Konkol and M. Konopík, "Named entity recognition for highly inflectional languages: effects of various lemmatization and stemming approaches," in *International Conference on Text, Speech, and Dialogue*. Springer, 2014, pp. 267–274.

[13] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint many-task model: Growing a neural network for multiple nlp tasks," in *EMNLP*, 2017.

[14] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 2227–2237.

[3] https://github.com/ardakdemir/pyJNERDEP

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[16] W. Che, Y. Liu, Y. Wang, B. Zheng, and T. Liu, "Towards better ud parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation," *CoNLL 2018*, p. 55, 2018.

[17] B. Heinzerling and M. Strube, "Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation," *arXiv preprint arXiv:1906.01569*, 2019.

[18] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," *arXiv preprint arXiv:1901.11504*, 2019.

[19] M. Straka, J. Straková, and J. Hajič, "Czech text processing with contextual embeddings: Pos tagging, lemmatization, parsing and ner," in *International Conference on Text, Speech, and Dialogue*. Springer, 2019, pp. 137–150.

[20] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 1638–1649.

[21] S. Wu and M. Dredze, "Beto, bentz, becas: The surprising cross-lingual effectiveness of bert," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 833–844.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[24] D. Zeman, J. Hajič, M. Popel, M. Potthast, M. Straka, F. Ginter, J. Nivre, and S. Petrov, "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*. Brussels, Belgium: Association for Computational Linguistics, October 2018, pp. 1–21.

[25] P. Qi, T. Dozat, Y. Zhang, and C. D. Manning, "Universal dependency parsing from scratch," *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 160–170, 2018.

[26] T. Dozat and C. D. Manning, "Simpler but more accurate semantic dependency parsing," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 484–490.

[27] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, 2015, pp. 2377–2385.

[28] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1064–1074.

[29] G. Tür, D. Hakkani-Tür, and K. Oflazer, "A statistical information extraction system for turkish," *Natural Language Engineering*, vol. 9, no. 2, pp. 181–210, 2003.

[30] G. Szarvas, R. Farkas, L. Felföldi, A. Kocsor, and J. Csirik, "A highly accurate named entity corpus for hungarian." in *LREC*, 2006, pp. 1957–1960.

[31] M. Ševčíková, Z. Žabokrtský, and O. Krůza, "Named entities in Czech: annotating data and developing NE tagger," in *International Conference on Text, Speech and Dialogue*. Springer, 2007, pp. 188–195.

[32] T. Ruokolainen, P. Kauppinen, M. Silfverberg, and K. Lindén, "A finnish news corpus for named entity recognition," *Language Resources and Evaluation*, pp. 1–26, 2019.

[33] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning (JMLR)*. Jmlr, 2013.

[34] Y. Goldberg, "Assessing bert's syntactic abilities," *arXiv preprint arXiv:1901.05287*, 2019.

[35] A. Akdemir and T. Güngör, "Joint learning of named entity recognition and dependency parsing using separate datasets," *Computación y Sistemas*, vol. 23, no. 3, 2019.

[36] G. A. Şeker and G. Eryiğit, "Initial explorations on using crfs for turkish named entity recognition," *Proceedings of COLING 2012*, pp. 2459–2474, 2012.

[37] A. Güneş and A. C. TantuĞ, "Turkish named entity recognition with deep learning," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2018, pp. 1–4.

[38] Y.-J. Chu, "On the shortest arborescence of a directed graph," *Scientia Sinica*, vol. 14, pp. 1396–1400, 1965.

[39] J. Edmonds, "Optimum branchings," *Journal of Research of the national Bureau of Standards B*, vol. 71, no. 4, pp. 233–240, 1967.

[40] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[41] T. Raiko, H. Valpola, and Y. LeCun, "Deep learning made easier by linear transformations in perceptrons," in *Artificial intelligence and statistics*, 2012, pp. 924–932.

[42] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[43] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[44] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, "The natural language decathlon: Multitask learning as question answering," *arXiv preprint arXiv:1806.08730*, 2018.

[45] E. Bejček, J. Panevová, J. Popelka, P. Straňák, M. Ševčíková, J. Štěpánek, and Z. Žabokrtskỳ, "Prague dependency treebank 2.5–a revisited version of pdt 2.0," in *Proceedings of COLING 2012*, 2012, pp. 231–246.

[46] V. Vincze, D. Szauter, A. Almási, G. Móra, Z. Alexin, and J. Csirik, "Hungarian dependency treebank," 2010.

[47] U. Sulubacak, M. Gökırmak, F. Tyers, Ç. Çöltekin, J. Nivre, and G. Eryiğit, "Universal dependencies for turkish," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 3444–3454.