IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 13, 2020, accepted October 22, 2020, date of publication October 27, 2020, date of current version November 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3033979

Part-of-Speech Tagging Using Multiview Learning

KYUNGTAE LIM[1] **AND JUNGYEUL PARK**[2]
[1]Intelligent Computing Laboratory, Korea Atomic Energy Research Institute, Daejeon 34057, South Korea
[2]Department of Linguistics, University of Washington, Seattle, WA 98195-2425, USA

Corresponding author: Jungyeul Park (jungyeul@uw.edu)

The work of KyungTae Lim was supported in part by the Korea Atomic Energy Research Institute through the project titled Enhancement of Operation Stability of HANARO Using Intelligent Computing Technology under Grant 524170-20.

ABSTRACT In natural language processing, character-level representations are vector representations of the particular character. Character-level representations have recently focused on enriching subword information by stacking deep neural models. Ideally, applications of several character-level representations can help capture different aspects of the subword information. However, this approach has often failed in the past, mainly because of the nature of traditionally used simple concatenation models. In this study, we explore different character-level modeling techniques. During the learning process, long short-term memory-based character representations can introduce different views for a part-of-speech tagger. After investigating two previously reported techniques, we propose two additional extended methods: (1) a multihead-attention character-level representation for capturing several aspects of subword information, and (2) an optimal structure for training two different character-level embeddings based on joint learning. We evaluate our results on the part-of-speech (POS) tagging dataset of the Conference on Natural Language Learning (CoNLL) 2018 shared task in universal dependencies. We show that our method substantially improves POS tagging results for many morphologically rich languages where the character information should be considered more substantially. Moreover, we compare the performance of our model with recent state-of-the-art POS taggers, which are trained with language models such as Bidirectional Encoder Representations from Transformers (BERT) and Deep Contextualized Word Representations (ELMo); our multiview tagger shows better results for nine languages. The proposed character model shows significant improvements in Ancient Greek, with average gains of 8.89 points in accuracy compared to the previous word representation model. Therefore, our empirical experiments indicate that character-level representations are more important than word representations for morphologically rich languages in terms of performance.

INDEX TERMS Part-of-speech tagging, multiview learning, character-level representation, neural networks, natural language processing.

I. INTRODUCTION

Natural language processing (NLP) has been focused on English and a few other languages that were economically (or more rarely, strategically) profitable. The gradual development of the Web and of social media has revealed the need to deal with more languages, which, in turn, have introduced new technological challenges. For example, languages exhibit a large diversity of morphological complexity, and NLP tools must tackle this diversity to obtain acceptable performances beyond English (*e.g.*, on agglutinative or polysynthetic languages). In this context, feature representation methods that capture morphological complexities have become an essential element of NLP systems.

The associate editor coordinating the review of this manuscript and approving it for publication was Seyedali Mirjalili.

In NLP, part-of-speech (POS) tagging is a token classification task that predicts the POS of each word in context. For example, given a sentence *the fox jumps*, a successful POS tagger can predict *the* as a determiner, *fox* as a noun, and *jumps* as a verb. POS tagging is a convenient approach to evaluate a system that has to deal with morphological diversity.

Recently, words have been commonly represented as vectors. Word embeddings provide fine-grained token representation that can help develop accurate taggers. However, word-level representations that focus only on word forms have two serious limitations:

- **Lack of subword information**: Word-level representations cannot capture subword information effectively because they focus only on word forms. Humans may

capture the following subword information from a word *delexicalized*: (1) *de* is used to indicate privation, removal, and separation; (2) *ed* represents the tense of the verb. However, word-level representations ignore these subwords.

- **Out-of-vocabulary words**: Out-of-vocabulary words are unknown because they were not presented in the training dataset. Word-level representations without external lexical resources (*e.g.*, pretrained embedding) are weak to handle these words. For example, if *delexicalized* is not part of the training corpus, the system cannot determine its deep context during POS tagging.

To address aforementioned problems, we represent words in multiple views, i.e., by using multiple features that represent a word form. For example, multiview learning in NLP may use the following multiple views: a token can be represented at the character level. It would be additionally augmented using stem, prefix, and suffix segments [1]. The utilization of multiview data has resulted in considerable success in various NLP problems. Combining different word representations at the character, token, or subword levels has proven to be helpful for dependency parsing [2]–[4] and other NLP tasks as well as POS tagging [5]–[7].

Additional views for word representations may be based on external resources such as external word embeddings (*e.g.*, Wor2Vec) or a language model (LM). In general, word embeddings trained on massive raw texts (*e.g.*, Wikipedia or OpenCrawl) can deliver more information than randomly initialized word embeddings during the learning process. This type of external word embedding may be beneficial for handling out-of-vocabulary words. Recently, systems trained with LM representations have shown state-of-the-art performances in NLP. ELMo [8], inter alia, which is trained to obtain unsupervised textual representations using bidirectional long short-term memory (BiLSTM), shows the best performance in the 2018 CoNLL shared task in POS tagging [1], [9]. Another cutting-edge LM, BERT [10], trained by bidirectional transformers with a masked language model strategy, shows outstanding results in POS tagging and dependency parsing [11].

Given multiple views, a simple yet popular approach has been to unify multiple representations into one. For instance, the best-performing team of the 2018 CoNLL shared task [9] concatenates character and word embeddings before contextualizing them via LSTM. This approach is particularly popular for neural networks because it is a direct method to concatenate multiple representations. All aforementioned studies have used this approach. However, the major problem of a simple concatenation of input vectors may lead to overfitting: the model may ignore specific statistical properties of each view [12]. In contrast, recently, meta-LSTMs [13] proposed extending the naive solution of concatenating representations in combination with building independent models for each view in the context of POS tagging. Meta-LSTMs build a meta single-view model on top of the concatenated multiview output while training all multiview models and the unified single-view model jointly. However, there still have not been significant studies on applying several character models simultaneously. This is an important issue for the recent NLP problems because there are several BERT-like models proposed, which use subword representations. Ideally, the applications of several character and subword-level representations can help to capture the different aspects of morpheme-like information. This approach has often failed in the past because of the nature of traditionally used simple concatenation models. To address this issue, one of the most promising approaches is to adapt meta-LSTMs on top of several subword models. Following on the previously proposed joint methods that promote the multiview approach with meta-LSTMs [14], [15], we propose a joint learning method for the better application of several subword models. This method can be trained using two novel lexical representations:

- **Multiattentive character-level representation**: A context-sensitive character-level representation that considers several aspects of subword information.
- **LM representation**: A deep contextualized representation that integrates richly supplied contexts gathered from external resources (raw texts) by using ELMo [8] and BERT [10].

We propose a new approach that is designed to offer a more complete representations of words. We use joint training techniques to induce the proposed character embeddings that focus on different aspects of subword information. This new technique has the advantage of capturing locally optimized character views (features) and globally optimized views regardless of the language type. We investigate how additional contextualized representations produce syntactically consistent tagging outputs. We test our multiview approach through POS tagging. The aim is to determine whether our contextualized representation is helpful for tagging. We report our results for the CoNLL 2018 shared task (ST) dataset using this tagger. In summary, our main contributions to the multiview POS tagger are three-fold:

1) We propose a SOTA POS tagger that considers several aspects of subword information.
2) We analyze the impact of the proposed representations and show the performance differences based on language types.
3) We explore different supervised learning scenarios where the size of training data varies with diverse morphologically rich languages.

The proposed character model can be applied most NLP systems that need to handle unknown words; further, the proposed taggers can be used for an additional prepossessing task to enhance morphological information. For further applications of our tagger, we have made the tagger publicly available at `https://github.com/jujbob/Utagger`.

The remainder of this article is structured as follows. Sections II–III present two different character embeddings, and Section IV details how these embeddings are combined.

The experiment and results are discussed in Section V. Finally, a conclusion is provided in Section VII.

II. MULTIATTENTIVE CHARACTER-LEVEL REPRESENTATIONS

Character-level representation methods have become a crucial component that provided better feature representations for many NLP tasks [16]–[18]. Character-level representations (character embeddings) are vector representations of a particular character. Word embeddings are trained using a sequence of words, whereas character embeddings are trained using a sequence of characters. As character embeddings utilize parts of words, they can capture subword information in a token or a sentence. Owing to these subwords, it is easier to represent common prefixes and suffixes with which an inflection expresses grammatical categories. For example, a word can have multiple forms, making it harder to find and look up words in the dictionary. By default, NLP systems have no rules to normalize or analyze unknown word forms. Thus, subwords can be well-suited for morphologically rich languages where the inflected forms of a word can vary because of the characteristics of some languages such as agglutination [19], [20].

Traditionally, a character-level word representation learned using LSTMs takes a sequence of characters as input and returns an encoded vector [16]. Recently, studies on character models have focused on enriching feature representations by stacking more neural layers [21], applying an attention mechanism [17], and appending a multilayer perceptron (MLP) to the output of recurrent networks [7]. This approach has obtained the best performance for POS tagging and dependency parsing in CoNLL 2017 and 2018 ST datasets [22], [23]. However, despite their benefits, most of these systems have clear shortcomings caused by their lack of representation of unknown words. Moreover, the application of several character models, which are capable of capturing different lexical characteristics, has not been fully explored so far. This is because two character models (such as two LSTM-based character representations) are often learned separately, and they generally capture almost identical features; therefore, they do not have a real positive influence on the results.

In this article, we propose a novel approach that can offer more complete word representations by using two complementary approaches: (1) a subword analysis geared towards recognizing morpheme-like information and (2) a contextual model that considers the sentential framing of a word, which is particularly useful for the analysis of unknown words where contextual information is sometimes sufficient to predict the meaning of an unknown word accurately. To achieve this, we combine two different character embeddings: a context-independent word-based character representation [21] and a context-sensitive sentence-based character representation [24], [25].

A. BASIC NOTION OF THE CHARACTER MODEL

Given an input sentence s of the character length n with the sequence of characters $s = (ch_1, \cdots, ch_n)$, our system creates a sequence of sentence-based character embeddings $ch^s_{1:n}$, which are initially encoded as random vectors. Because a sentence s is composed of m words such that $s = (w_1, \cdots, w_m)$ and each word w_i can be decomposed into a sequence of characters $w_i = (ch^i_1, \cdots, ch^i_k)$ where k is the length of w_i, the system creates a set of sequences of word-based character embeddings $ch^1_{1:k_1}, \cdots, ch^m_{1:k_m}$.

A character is represented by two different embeddings: from a sequence of characters in the sentence $ch^s_{1:n}$, and from a sequence of characters in the i-th word $ch^i_{1:k}$. These two embeddings are different because they are initialized randomly. We use lowercase italics for vectors and uppercase italics for matrices where w and W denote parameters that the system must learn. Moreover, a semicolon denotes the concatenation of two vectors and *ConcatRow* represents the concatenation of matrices by rows. We maintain this convention when indexing and stacking: h_i is the i-th vector of matrix H, and matrix H is the stack of all vectors h_i.

B. CONTEXT INSENSITIVE WORD-BASED CHARACTER MODEL

A sentence comprises a sequence of characters; likewise, a word also comprises a sequence of characters. Therefore, there are at least two approaches for building character-level word embeddings using LSTM. Let us call a word-based character model a method that generates a representation based on a word (i.e., not the full sentence). In general, word-based character vectors are obtained by running a BiLSTM [7] over the k characters $ch^i_{1:k}$ of a word w_i as

$$f^{(wc)}_j, b^{(wc)}_j = BiLSTM(r^{(wc)}_0, (ch^i_1, .., ch^i_k))_j$$
$$h^{(wc)}_j = [f^{(wc)}_j; b^{(wc)}_j]$$

where $f^{(wc)}_j$, $b^{(wc)}_j$, and $h^{(wc)}_j$ denote the forward-pass hidden layer of the BiLSTM for character j of a word, backward pass, and concatenation of the two, respectively. Previous studies have shown that the last encoded character $f^{(wc)}_k$ represents a summary of all the information from the input character sequence [21] because the output of the LSTM tracks the contextual information. However, taking the last encoded character as an embedding is often omitted in the earlier parts of the sequence once the entire sequence has been processed. This is because the last encoded embedding $f^{(wc)}_k$ is a vector of a fixed size that becomes a bottleneck when summarizing long sequences into a single vector. The attention mechanism was invented to address this problem. The idea of attention is to use all intermediate states $H^{(wc)}_i$ to build the context vector. An attention method that establishes relations between different parts of a single sequence to extract a specific representation is called self-attention. The self-attention method involves applying a linear transformation [17] over the matrix

of character encodings $H_i^{(wc)} = h_{1:k}^{(wc)}$, for which attention weights $a_i^{(wc)}$ are calculated as

$$a_i^{(wc)} = Softmax(w^{(wc)} H_i^{(wc)})$$
$$c_i^{(wc)} = a_i^{(wc)} H_i^{(wc)}$$

where $w^{(wc)}$ denotes a linear transformation parameter. The self-attention weight $a_i^{(wc)}$ intuitively corresponds to the most informative characters of word w_i for the task being learned. Passing the encoded character vector $H_i^{(wc)}$ of each word through its attention weights $a_i^{(wc)}$, we obtain the character-level word-vector as $c_i^{(wc)}$. Figure 1 is an illustration of the word-based character model including the attention weights $a_i^{(wc)}$ and the encoded character vector $H_i^{(wc)}$. Dozat *et al*. [7] suggested concatenating the encoded vector $f_k^{(wc)}$ with the attention vector $a_i^{(wc)} H_i^{(wc)}$ to capture both the summary and subword information as a unified representation for POS tagging (Figure 3-(A)). However, as reported by Lin *et al*. [26], self-attention based representations tend to focus on a specific component of the sequence. To alleviate this, we propose multihead-attention character-level word embeddings that reflect various character-level features of a word by applying multiple attention weights described as a matrix $A_i^{(wc)}$ rather than a vector $a_i^{(wc)}$.

$$A_i^{(wc)} = Softmax(W^{(wc)} \tanh(D^{(wc)} H_i^{(wc)}))$$
$$c_i^{(wc)} = ConcatRow(A_i^{(wc)} H_i^{(wc)}) \quad (1)$$

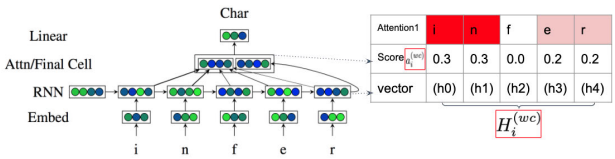

FIGURE 1. Example of a word-based character model with a single attention representation [7].

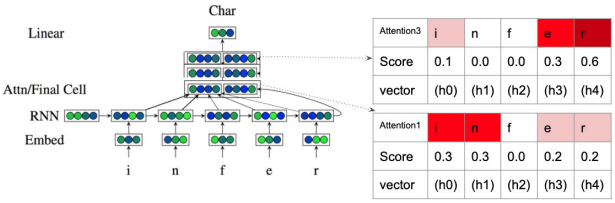

FIGURE 2. Example of a word-based character model with three attention representations.

By applying an attention parameter matrix $W^{(wc)}$ rather than a vector $w^{(wc)}$ and a nonlinear function with a weight parameter $D^{(wc)}$, the attention weight can reflect several aspects of subword information. For example, a successfully trained attention weight $W^{(wc)}$ can recognize two different morphemes: *in* and *er* from the word *infer* as presented in Figure 2. However, we cannot determine which subword will be captured during the training because the system only attempts to adjust the attention parameter W based on the prediction errors [26].

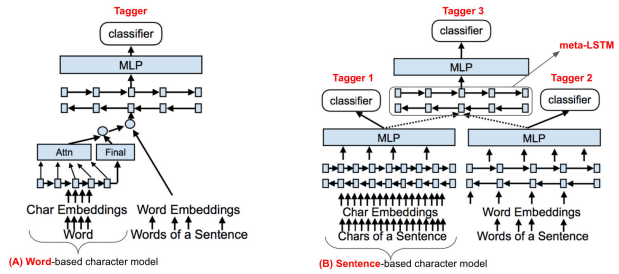

FIGURE 3. (A) Structure of the tagger proposed by Dozat *et al*. [7] using a word-based character model. (B) Structure of the tagger proposed by Bohnet *et al*. [25] using a sentence-based character model with meta-LSTM.

C. CONTEXT-SENSITIVE SENTENCE-BASED CHARACTER MODEL

The model described previously is effective at capturing subword information. However, it cannot capture contextual information beyond word boundaries because the model takes only a word as input, and it cannot consider the context of this word. The overall context can be modelled by considering the entire sentence as a sequence of characters. The sentence-based character model is a model that builds a word representation based on the sequence of corresponding characters in a sentence. Thus, the sentence-based character model is a context-sensitive model [25].

Alberti *et al*. [24] used a sentence-based character representation trained by an LSTM model for dependency parsing and achieved SOTA POS tagging results for morphologically rich languages. In POS tagging, Bohnet *et al*. [25] showed that a sentence-based character model that processes all characters of a sentence at once is better at retaining context information for unseen data compared to that using a token-based model. Figure 3-(B) shows the structure of sentence-based model that obtained the best POS tagging results for the CoNLL 2018 ST.

We propose extending these previous approaches with a self-attention sentence-based word embedding, which is composed of three parts:

1) **Encoding**: A single encoding is generated from the entire sequence of characters corresponding to a sentence using a BiLSTM as

$$f_j^{(sc)}, b_j^{(sc)} = BiLSTM(r_0^{(sc)}, (ch_1^s, .., ch_n^s))_j$$
$$h_j^{(sc)} = [f_j^{(sc)}; b_j^{(sc)}]$$

2) **Slicing**: The output of the BiLSTM is sliced from the start index $sidx(w_i)$ to the end index $eidx(w_i)$ of each word. A matrix $H_i^{(sc)}$ is produced by stacking the encoded character vectors of a word as

$$H_i^{(sc)} = (h_{sidx(w_i)}^{(sc)}, .., h_{eidx(w_i)}^{(sc)})$$

3) **Attention**: $H_i^{(sc)}$ is transformed into the multihead-attention representation $c_i^{(sc)}$, as with the word-based model in (1).

$$A_i^{(sc)} = Softmax(W^{(sc)} \tanh(D^{(sc)} H_i^{(sc)}))$$
$$c_i^{(sc)} = ConcatRow(A_i^{(sc)} H_i^{(sc)}) \quad (2)$$

Our approach is distinct from Bohent *et al*. [25] who proposed the concatenation of only the first and the last BiLSTM outputs as $c_i^{(sc)} = MLP([h_{sidx(w_i)}^{(sc)}; h_{eidx(w_i)}^{(sc)}])$. Their concatenation method can capture a summary of all information contained in the input character sequence. However, it can be a bottleneck as explained in Section II-B. To rectify this problem, we adopt the multihead-attention model described in the previous section with $H_i^{(sc)}$. We believe that this adopted multiattention model is more accurate in capturing the context.

III. DEEP CONTEXTUALIZED LANGUAGE MODELS

The meaning of words changes according to the context. However, traditional pretrained word embeddings do not consider the context that can be farmed from a sentence. To address this problem, we need to model the entire sentence as a source of contextual information. This is the goal of deep contextualized word embeddings that aim to model word semantics according to the context. Representing the sentence via an LSTM is a popular method to transform a word vector into a contextualized representation because of its memory cells. Embeddings from language models (ELMo) are deep contextualized word representations designed to model complex characteristics of word use such as syntax and semantics [8]. These word embeddings are learned functions of the internal states of LSTM pretrained on a large text corpus. Contextualized embeddings is a new technique for word representation that has achieved SOTA performance across a wide range of language understanding tasks; this approach can capture both subword and contextual information. From a technical point of view, ELMo can be used as a function that provides a deep contextualized word representation based on the entire input sentence using a pretrained LSTM. Recently, BERT [10] trained by bidirectional transformers with a masked language model strategy showed good results in tagging and parsing [11], [15], [27]. As an external word view (representation), we use both ELMo and BERT representations on our tagger. We report the effect of these views in Section V.

IV. DEEP CONTEXTUALIZED MULTIVIEW POS TAGGER

In this section, we describe our POS tagger, which uses joint many-task learning (JMT) to integrate the contextualized representations. The JMT is a subfield of machine learning in which multiple learning tasks are solved simultaneously. JMT enriches context-sensitive feature representations by learning different tasks with shared parameters [28]. For example, we can train a tagger and a parser simultaneously with shared word representations. Further, this approach may consist of training several classifiers for the same task. For example, Figure 3-(B) shows the structure of a meta-tagger that applies a JMT approach to a single POS tagging task using a meta-LSTM [25]. The meta-tagger separately trains a word-based and a character-based POS tagger without sharing parameters, and it then joins the two models via another middle-layer BiLSTM (meta-LSTM) and MLP, which is used as the final classifier. Hence, the system integrates three different taggers trained from different views: character, word, and character + word. However, the concatenation method that concatenates the character and word features before contextualizing them (see Figure 3-(A)) has been the most popular approach recently. This model generally has only one tagger.

The concatenation method often tends to rely on a specific feature (*e.g.*, word) and ignores the other features (*e.g.*, character) during training [26]. The meta-tagger offers the advantage of learning contextual information because each tagger attempts to identify the best representations within the limited character and word features. Because the middle-layer BiLSTM has access to two different contexts captured by two different taggers, we refer to it as a meta-BiLSTM. The meta-BiLSTM approach is particularly effective at balancing the application of word and character-based features.

As each separated tagger is trained on an individual classifier, each tagger in previous work struggled to identify the best features. We train two character-level taggers and combine them through a meta-BiLSTM POS tagger. As mentioned above, we use two different sentence and word-based character embeddings rather than only one ultimate embedding. While Bohnet *et al*. [25] also uses two embeddings in a similar way, we use the word-based character embedding instead of the simple word embedding. Thus, we can take advantage of both the meta-BiLSTM model and JMT because we are then using two character-level models. Figure 4 shows the overall structure of our contextualized tagger.

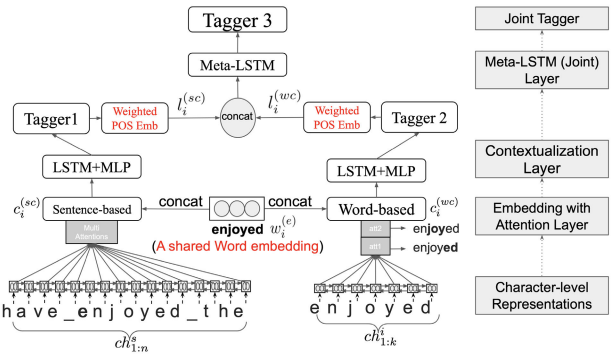

FIGURE 4. Overall structure of the proposed contextualized tagger with three different classifiers.

A. TWO TAGGERS FROM CHARACTER MODELS

To take advantage of the meta-LSTM, our system builds two POS taggers using the two different word embeddings generated from the sentence c_i^{sc} in (2) and the word-based c_i^{wc} in (1) character models, as described in Section II. To enrich word-level information for each character model, the system concatenates a shared word embedding $w_i^{(e)}$ initialized with a pretrained word embedding [29]. Further, this is concatenated with ELMo ($e_i^{(el)}$) or BERT ($e_i^{(be)}$) embeddings if available. These embeddings go through another BiLSTM layer whose outputs are $g_i^{(sc)}$ and $g_i^{(wc)}$ for sentence-based and word-based

representations, respectively. Finally, we apply an MLP classifier with a weight parameter $Q^{(sc)}$ including a bias term $b^{(sc)}$ to classify the candidate POS label as

$$\begin{aligned} p_i^{(sc)} &= Q^{(sc)} MLP(g_i^{(sc)}) + b^{(sc)} \\ y_i^{(sc)} &= \arg\max_j p_{i,j}^{(sc)} \end{aligned} \tag{3}$$

$$\begin{aligned} p_i^{(wc)} &= Q^{(wc)} MLP(g_i^{(wc)}) + b^{(wc)} \\ y_i^{(wc)} &= \arg\max_j p_{i,j}^{(wc)} \end{aligned} \tag{4}$$

Performing the same operation for the word-based embeddings, we predict two POS tags for $y_i^{(sc)}$ and $y_i^{(wc)}$.

B. JOINT POS TAGGER

To create joint representations that learn how to combine the output of the two taggers, we transform the two tagging results as a weighted POS label embedding $h_i^{(pos)}$ as

$$\begin{aligned} l_i^{(sc)} &= \sum_{j=1}^{U} P(y_i^{(sc)} = j | p_i^{(sc)}) pos(j) \\ h_i^{(pos)} &= [l_i^{(sc)}; l_i^{(wc)}] \end{aligned} \tag{5}$$

where U, $P(y_i^{(sc)} = j|p_i^{(sc)})$, and $pos(j)$ denote the number of possible POS tags, probability that the j-th POS tag is assigned to a word w_i, and a randomly initialized POS vector, respectively.

For example, Figure 5 shows a weighted POS embedding for a token p_i. A weighted POS embedding for p_i is the weighted sum of the POS embedding based on its probability $P(y_i^{(sc)} = j|p_i^{(sc)})$. We generate a joint embedding $h_i^{(pos)}$ by concatenating two weighted POS features [$l_i^{(sc)}$ and $l_i^{(wc)}$]. With the classifier proposed in (3) that takes the joint vector $h_i^{(pos)}$ as an input, the system predicts another POS tag $y_i^{(pos)}$ from two weighted POS features. Note that the system only uses tokenized sentences as input and the weighted POS results are predicted at the training time.

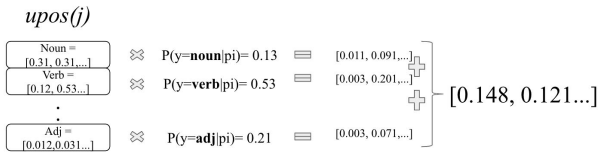

FIGURE 5. Example of a procedure to generate a weighted POS embedding.

A meta-tagger [25] trains three taggers using separate optimizations. We train our taggers simultaneously using a single Adam optimization algorithm [30], with a summed cross-entropy loss for each tagger. This approach has the advantage of passing error propagation directly to the shared word embedding $w^{(e)}$.

V. EXPERIMENTS AND RESULTS

In this section, we present our experimental settings and the results of our model on the CoNLL 2018 ST dataset. The objective of the CoNLL 2018 ST to evaluate POS tagging and dependency parsing by following a real-world setting that starts from raw texts over 57 languages. Further, the task was intended to evaluate the results for many typologically different languages. Therefore, this dataset is useful for estimating the performance of our system over typologically different languages. We evaluate our approach on 12 test treebanks from 8 different languages with different types of writing systems to investigate the effect on the various languages of our approach. The languages we have investigated are presented in Table 2. We compare our universal part-of-speech (UPOS) tagging results with the official CoNLL 2018 results.[1]

A. DATASET

We evaluate our model on the Universal Dependency (UD) 2.2 corpora provided for the CoNLL 2018 ST [31]. For a fair comparison of our experiments with the official 2018 CoNLL results, we use the pretrained word embeddings for Japanese and Chinese[2] provided by the CoNLL organizer, and word embeddings for other languages described in [29]. Further, we use ELMo embeddings trained by Lim *et al*. [1].

The input of the shared task is usually given as raw texts. The tokenization results of the best-performing team have been officially provided by the shared task organizer after the shared task.[3] For our input test data, we use the tokenization results from the best-performing models of each treebank in the shared task to compare the POS tagging result directly. When the baseline tokenizer [32] is used for preprocessing, it largely affects the results because tokenization has a direct and massive effect on tagging performance. Whereas five treebanks in the test dataset have no training data available, at least one large training treebank in a different domain for each language, instead for training, is available. This setting offers an excellent opportunity to explore the ability of our joint character model to deal with unseen data. Moreover, BERT has been trained by a sentence-based subword model using external resources (*e.g.*, word piece model [10]). This allows us to investigate three different character models effectively for unknown words.

B. EXPERIMENTAL SETUP

We apply the same hyperparameter settings as in Smith *et al*. [33] for BiLSTM dimensions, MLP, optimizer including β, and the learning rate to compare our results with them in a similar environment. Table 1 presents our hyperparameter settings. We set 300 dimensions for parameters W and D in (1) and Q in (3). The same dimensionality is applied to the sentence-based character model and the word-based model. During training, we run over the entire training data as an epoch with a batch size of 32 randomly selected sentences. We save the model with the best performance on the development set within 300 epochs.

[1] http://universaldependencies.org/conll18/results-upos.html
[2] http://hdl.handle.net/11234/1-1989
[3] http://hdl.handle.net/11234/1-2885

TABLE 1. Hyperparameters.

Component	Value
ch_i (char) dim.	100
$h_i^{(pos)}$ (weighed POS) dim.	200
W, Q, D (parameters) dim.	300
$e_i^{(el)}$ (ELMo) dim.	1024
$e_i^{(be)}$ (BERT) dim.	768
$h^{(wc)}$ (word-based) output dim.	300
$h^{(sc)}$ (sentence-based) output dim.	300
No. BiLSTM layers	2
MLP output dim.	300
Dropout	0.3
Learning rate	0.002
β_1, β_2	0.9, 0.99
Epoch	300
Batch size	32
Gradient clipping	5.0

The proposed neural model (Join+Roberta) uses around 15.3 GB GPU memory when training on training data with a batch size of 32. Under this condition, a total of 330M parameters is required including 270M parameters by the RoBERTa model. During training, each epoch runs around 25 min over the training data. The proposed neural model approximately handles 131 tokens per second. As our model needs to consider the number of words and characters as the input sequence, the time complexity is $O(n^2)$. However, applying two character-level representations require approximately 7 to 11 percent of extra GPU memory than models that only use a character representation. To find optimal hyperparameter values, we apply different batch sizes, hidden dimensions, and learning rates; however, we cannot find meaningful differences from the settings proposed by Smith *et al*. [33].

C. RESULTS

Table 2 summarizes the results for the test sets of each treebank compared with the best performance `winn` as reported in the official ST results. The `join` column represents our model jointly trained by the two different character-level representations described in Section IV-B; the `joinE` column is the `join` model enhanced with ELMo embeddings.

Overall, our `joinE` model achieves SOTA results compared with the official results published for the ST except for the case of en_pud where the best-performing model applied both ELMo and an ensemble of different models [1]. Even without the application of ELMo, our `join` model shows comparable results with the models that used ELMo embeddings (marked *) [9] and an ensemble (marked $^+$) [1].

Table 2 indicates that while the last five test treebanks have no training data, large treebanks exist in different domains for these languages. For example, the en_pud (genre: news) test treebank does not have a specific training treebank. However, it has alternative training treebanks from a different domain such as en_ewt (genre: blog, reviews). Hence, we can investigate the extent to which our joint character model is helpful for handling unseen data. We test those five treebanks of parallel universal dependencies (PUD) with models trained on other training corpora (en_ewt, ja_gsd, cs_pdt, sv_talbanken, and fi_tdt) for a fair comparison with results presented in the shared task [1], [33]. We observe that the proposed model can handle cross-domain data in a manner comparable to other systems on the PUD treebanks even though we do not apply ELMo embeddings for Czech, Swedish, and Finnish (`winn` does not use ELMo embeddings for these languages).

There is an interesting observation based on the number of training sentences. If the training dataset is relatively small as for sv_lines, en_gum, and en_lines (less than 3000 sentences), we observe more positive gains than the model that uses a larger treebank such as cs_pdt (68,495 sentences). Hence, our joint model (based on two-character models) can capture better contextual features even in less-resourced scenarios.

1) EFFECT OF THE JOINT LEARNING MECHANISM

To investigate whether our joint model yields an actual benefit compared to a simple concatenation approach, we test a

TABLE 2. Universal part-of-speech (UPOS) tagging results compared with the best-performing team (`winn`) of each treebank for the ST. Size denotes the size of the training corpus. The included models are our joint model (`join`), joint with ELMo (`joinE`), concatenated (`conc`), and ELMo only (`ELMo`). Symbols * and $^+$ represent the solutions that applied ELMo embeddings or an ensemble, respectively. For Czech, Swedish, and Finnish, we could not find the ELMo embedding that has been used by the best-performing team. sv_li+ta represents the concatenation of sv_talbanken and sv_lines.

Test set		Train set	Size	winn	joinE	elmo	join	conc
zh_gsd	(Chinese)	zh_gsd	3997	91.94*	**93.29**	92.47	91.97	91.81
ja_gsd	(Japanese)	ja_gsd	7164	92.97*	**93.12**	93.01	92.99	92.83
en_ewt	(English)	en_ewt	12543	95.94^{*+}	**95.99**	95.81	95.65	95.39
en_lines		en_lines	1022	97.06^{*+}	**97.45**	96.76	96.57	96.39
en_gum		en_gum	981	96.44^{*+}	**96.45**	96.25	96.08	95.95
fr_gsd	(French)	fr_gsd	14554	96.97	**97.18**	97.04	97.04	96.85
ko_gsd	(Korean)	ko_gsd	27410	96.33*	**96.58**	96.15	96.21	96.17
en_pud	(English)	en_ewt	0 (12543)	**96.21**$^{*+}$	96.08	96.17	95.90	95.78
ja_mod	(Japanese)	ja_gsd	0 (7125)	54.60	**54.70**	54.61	54.67	54.55
cs_pud	(Czech)	cs_pdt	0 (68495)	97.17	-	-	**97.21**	96.81
sv_pud	(Swedish)	sv_li+ta	0 (7479)	94.28$^+$	-	-	**94.29**	94.09
fi_pud	(Finnish)	fi_tdt	0 (12217)	97.65$^+$	-	-	**97.67**	97.50

disjoint model conc where a word embedding is defined simply as a concatenation of embeddings for different levels of representation.

$$h_i^{(pos)} = [c_i^{(sc)}; c_i^{(wc)}; w_i^{(e)}]$$

This model trains a single tagger without our joint learning method (Figure 3-(A)). As our empirical results demonstrate in Table 2, the join model always yields better performance than the conc model.

Notably, the shared word embedding $w^{(e)}$ produced by our joint learning procedure leads to consistent improvements both for the joint model and the two character-based models. Then, we use shared word embedding only for the sentence-based or word-based character model, and the performance decreases by an average of 0.05–0.20 absolute points over the three taggers. We observe almost identical results with the multitask learning approach for training a tagger and a parser simultaneously with or without shared embeddings [28].

2) EFFECT OF ELMo

Although the best-performing system [33] for UPOS tagging in ST outperformed the macro-average score over all treebanks with a 0.72 gap over the second-ranked team, some teams that used ELMo obtained the best score on many languages. Hence, it is necessary to extend the evaluation by using the same ELMo embedding as applied in the ST.[4] To investigate the performance of previously proposed taggers with ELMo, we evaluate the ELMo model that is a simple concatenation model with a single classifier presented in Figure 3-(A). This model produces a single tagger based on the concatenation of ELMo, word-based character, and word embeddings as

$$h_i^{(pos)} = [c_i^{(wc)}; e_i^{(el)}; w_i^{(e)}] \quad (6)$$

As indicated in Table 2, the ELMo model yields better performance than our join model but not for joinE. We found a relatively significant gap between ELMo and joinE in tagging Chinese and Korean, where English and French do not show such a gap. As reported by Smith *et al*. [34], we assume that languages that have a the large number of characters benefit more from the character embeddings, especially when embeddings are trained with external data.

There is some indication that ELMo is more influential than the dynamically trained character embeddings under less-resourced conditions. This is because ELMo is trained on various external resources. While our character models show relatively good results even if they are trained only on limited data without external resources, our models greatly benefit from using ELMo to learn deep contexts (See the size and performance gaps between join and joinE on zh_gsd).

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

3) EFFECT OF THE SELF-ATTENTION APPROACH

Table 3 demonstrates the effectiveness of the multihead-attention component for the word and sentence-based character models as expressed in (1). In Table 3, n represents the number of attention heads. The number of rows for attention heads is allocated to matrix A_i in (1), with additional rows providing traction for the model to focus on different semantic components of the word.

TABLE 3. Tagging results for eu_bdt (Basque) by the number of attention heads n of the word and sentence-based character embeddings. Here, word and sent denote the models with the taggers trained only on word- and sentence-based character representations (as described in (1)).

# of head	n=1	n=2	n=3	n=5
word-based char	96.04	96.17	**96.19**	96.12
sent-based char	96.24	**96.26**	96.21	96.17
join	96.39	96.40	**96.43**	96.35

We can observe at least marginal improvements when expanding from a single-head $n = 1$ to a double-head $n = 2$ for all models, and then to triple-head $n = 3$ for the word and join models. The applied model $n = 1$ is the popular single-head model proposed by Dozat *et al*. [7]. We observe a negative effect when expanding beyond five rows for all models. This is because, each additional attention-score $a_i^{(wc)}$ tends to focus on the same part of a sequence even though it requires an n-times higher-dimensional space [26].

4) ABLATION STUDY ON THE PROPOSED CHARACTER REPRESENTATION

In Section II, we described two different character representations and showed positive effects on tagging performances. The remaining question we need to investigate is how much performance gains can we expect on individual character representations trained on the different boundary of a sentence and a word? To analyze the effect of each character representation, we conduct a simple ablation study on the proposed character representation. The results are shown in Figure 6. We investigate Chinese UPOS tagging performance when varying the number of training samples used on three different character models. In the figure, join, word-based,

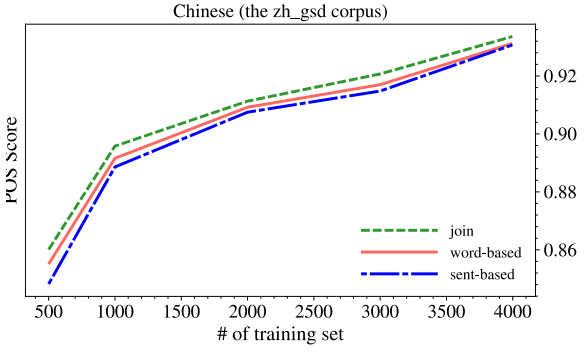

FIGURE 6. Evaluation results for Chinese (zh_gsd) based on the different sizes of the training set and the proposed character models.

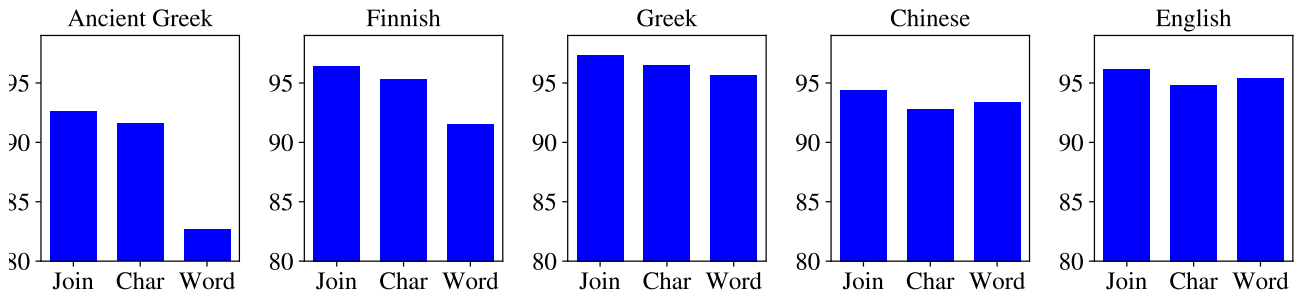

FIGURE 7. Evaluation results for five languages: Ancient Greek (grc_perseus), Finnish (fi_ftb), Greek (el_bdt), Chinese (zh_gsd), and English (en_ewt) based on different views. `join`, `char`, and `word` denote our joint model, sentence-based character model, and word embedding model, respectively.

and `sent-based` denote POS taggers trained with our word-based, sentence-based, and joint character representations, respectively. To observe performance gains for only character-level representations, we have not used any other embeddings (*e.g.*, word embedding and LMs). We find the following interesting results for this evaluation scenario. First, the performance of our `joint` model outperforms word or sentence-based models over all scenarios. Second, the word-based model shows slightly better results than the sentence-based model in low-resource scenarios with +0.87 points compared with the sentence-based character model when using only 500 training sentences. However, depending on the language, word and sentence-based character models show different performances. For example, the sentence-based character model always shows better performances for Basque (See Table 3). In contrast, the word-based model performs better for Chinese.

5) ABLATION STUDY ON THE CHARACTER AND WORD REPRESENTATION

We compared performance depending on the boundary of character-level representations. According to our results, the performance of these proposed character models can vary based on the language. As a further investigation, we want to understand the effect of each word and character representation for morphologically rich languages. The selected languages are Ancient Greek, (Modern) Greek, and Finnish as morphologically rich languages, and Chinese and English as counterexample languages. The morphologically rich languages include dozens of grammatical cases indicated by different affixes of varying lengths. In Figure 7, `join` denotes our joint model. Next, `char` is a model trained with only our sentence-based character representation as $h_i^{(pos)} = [c_i^{(sc)}]$, with a single attention head. Moreover, the `word` model is a model trained only with word embedding as $h_i^{(pos)} = [w_i^{(e)}]$. According to our results, the `char` model outperforms `word` for morphologically rich languages. The most surprising result we found is for Ancient Greek. There is a significant gap between `char` and `word` in Ancient Greek with 8.89 points, where Ancient Greek has a richer vowel system and grammatical features compared to Modern Greek. Almost the same gaps are observed on the Ancient Greek results from the CoNLL 2018 ST depending on whether a character model has been applied.[5] Further, Finnish follows the same performance patterns depending on the application of the character model. In contrast, the word embedding model shows a slightly better performance than the character model for Chinese and English. Based on these empirical experiments, we assume that character-level representations are more important for morphologically rich languages because of its ability to capture morphological information.

6) COMPARISON WITH SOTA TAGGERS TRAINED WITH LMs

Several BERT-like models that have been proposed in NLP show outstanding performances over almost all NLP tasks. In POS tagging, multilingual BERT[6] that can handle 100 languages is applied to train a multilingual POS tagger, namely, `udify` [11] in 2019. More recently, Lim *et al.* [15] proposed a `Co-meta` tagger that leverages its performance based on a semi-supervised learning approach with the multilingual BERT and BERT-base monolingual English model, and they achieved SOTA results. We compare the performance of our tagger with BERT-like models as existing SOTA systems, in particular, with multilingual BERT, BERT-base, and Roberta [35]. Table 4 shows the English POS tagging results of our tagger and other SOTA taggers. Note that we use gold tokenized results as our input to compare the results

[5]http://universaldependencies.org/conll18/results-upos.html
[6]https://github.com/google-research/bert/blob/master/multilingual.md

TABLE 4. UPOS for the English (en_ewt) corpus for each model, with BERT-like models.

Model	Char Model	LM	UPOS
`LATTICE` (2018) [1]	word-based	-	96.31
`udpipe` (2019) [11]	word-based	-	96.29
`join`	word-,sent-based	-	96.48
`LATTICE` (2018) [1]	word-based	`ELMo`	96.83
`join`	word,sent-based	`ELMo`	96.90
`udify` (2019) [11]	word-based	`bert-multi`	96.21
`Co-meta` (2020) [15]	sent-based	`bert-multi`	96.80
`join`	word,sent-based	`bert-multi`	**96.91**
`Co-meta` (2020) [15]	sent-based	`bert-base`	97.03
`join (our Work)`	word,sent-based	`bert-base`	**97.06**
`join (our Work)`	word,sent-based	`roberta`	**97.44**

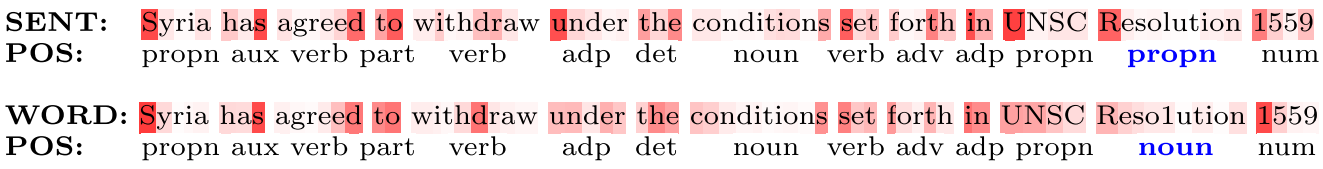

FIGURE 8. Visualization of attention maps generated by an input sentence and the corresponding attention weights based on the sentence-based (SENT) and word-based (WORD) models with the predicted POS. The depth of red color indicates the score of attention weights ($a_i^{(wc)}$, $a_i^{(sc)}$).

directly with the SOTA taggers because these taggers used gold tokenized results for UPOS tagging.

Before we deep dive into the results on BERT, we check the contribution of our character-level representations and meta-LSTM on tagging performance without LM as our baseline model. The first three rows of the table mention three taggers that do not use LM; namely `LATTICE` [1], `udpipe` [11], and our current `join` tagger. The `LATTICE` tagger showed the best performance in the ST in 2018 for English. They applied the word-based character model only. `udpipe` was the best-performing tagger without LM over more than 20 languages in 2019. `udpipe` proposed a multitask approach that trained a tagger and a parser simultaneously using shared LSTMs. Overall, our `join` model shows a systematically slightly higher score than `LATTICE` and `udpipe` by $+0.17 \sim 0.19$ points, respectively. Further, because the performance of English has already reached over 96% accuracy, there is no significant room for improvement. However, when we apply ELMo, the performance gaps between `LATTICE` and `join` decrease by $+0.07$ points (96.83 vs. 96.90). We assume that ELMo can bridge the gap between `join` (word and sentence-based character embeddings) and `LATTICE` (word-based character embedding).

Compared with taggers that used BERT-like models, our `join` model shows slightly better performances. Even though `Co-meta` applied both the meta-LSTM and the sentence-based character model [15], our `join` model that applies two character models showed higher performances. However, it should be noted that the `udfy` model was first trained with 75 different languages and then tuned for English [11]. Therefore, it is not clear how the multilingual training process affects the final performance for each language. Another interesting result is shown in the last row in Table 4 in which we apply the `join` model with `roberta`. We observe significant improvement, which is similar to that for other NLP systems which showed performance improvement when using `roberta` [35]. We conjecture that this is because, `roberta` is trained on more unlabeled data compared with `bert-base`, and thus, it can handle out-of-vocabulary words in a better manner.

VI. ANALYSIS AND DISCUSSION

A. ANALYSIS

In this section we report our analysis on the results. Further, summarize our findings in the depth of individual tag levels to investigate the detailed effect of the proposed feature representations with examples.

1) WHY DOES THE PROPOSED JOINT CHARACTER MODEL WORK?

One would naturally assume that a major reason for the performance improvements using two different character features would be larger feature dimensions because the sentence and word-based features are directly concatenated. The result of our ablation study suggest that the key factors contributing to the better performance of character representations are

1) Two different character features can make the model localize better individual words.
2) The sentence and word-based character features can capture slightly different aspects of subwords.

For an extensive analysis of how differently sentence and word character models work, we conduct a statistical investigation of its attention weights ($a_i^{(wc)}$, $a_i^{(sc)}$) as proposed in 1. We found that the sentence-based model is relatively sensitive to the first character of a word than the word-based model. The average attention weights on the first character over the entire test set is 25.47 for the sentence-based and 19.80 for the word-based models, respectively.

Figure 8 show a visualization result of a sentence in the test set with its attention weights. Overall, the word-based and the sentence-based character models focus on similar subwords. However, when observing only the first character of words, we find that the sentence-based model focuses more on the first character. Another interesting factor is that the sentence-based model is robust for capturing the continuous sequence of proper nouns. For example, *UNSC* (PROPN, a proper noun) and *Resolution* (PROPN) are proper nouns in Figure 8. Although the sentence-based model predicts correctly, the word-based one does not. We assume that this is because the sentence-based model is trained by a sentence-level LSTM; it can thus keep more context-sensitive information over a sentence than the word-based model. The word-based model seems to consider more grammatical information such as the tense of the sentence to focus on the subword *ed* from *agreed* and *s* from *has*. As we shown in Section V-C, applying joint learning for the sentence and the word-based model can help capture both local and global subword features. In contrast, using the same type of character models allows capturing almost identical features (*e.g.*, applying two word-based models). Thus, the effect of using

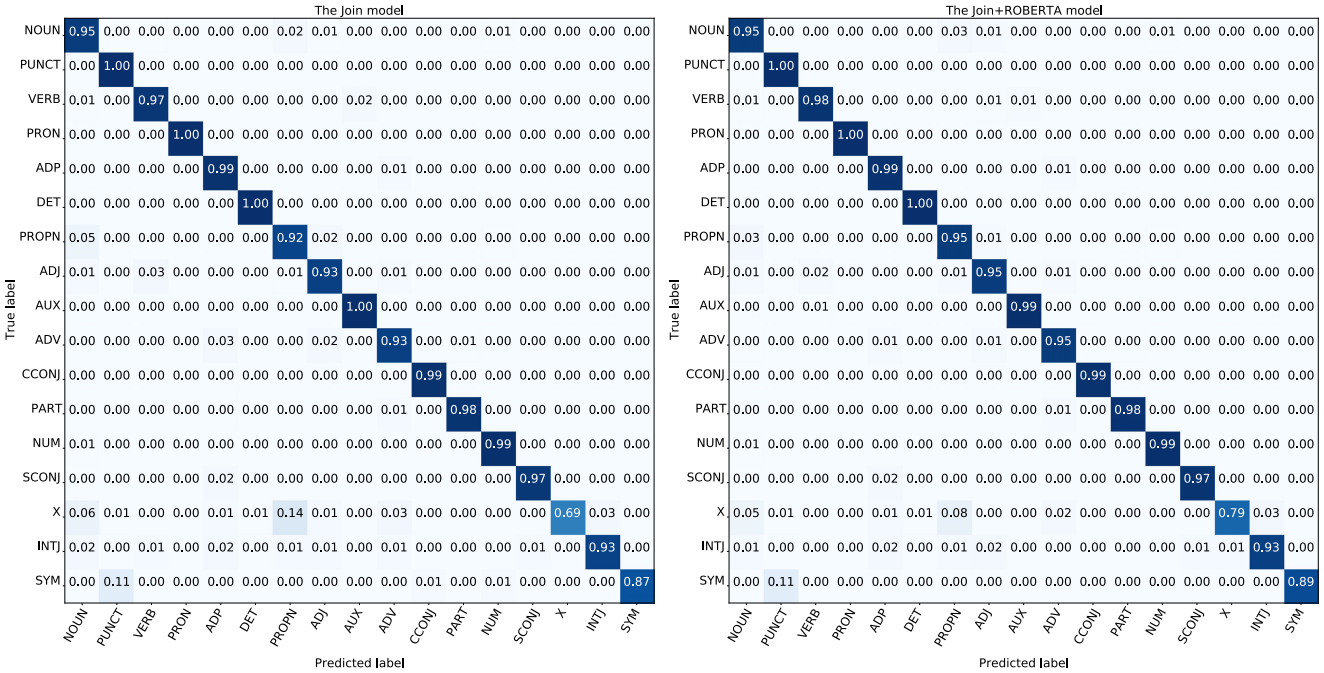

FIGURE 9. Confusion matrix of our `join` (left) and `join`+ROBERTA model. We found that applying ROBERTA helps distinguish NOUN (common noun) and PROPN (proper noun).

only the character features is relatively smaller, or sometimes, it leads to performance degradation.

2) WHY BERT-LIKE MODELS ALWAYS OUTPERFORM CHARACTER-ONLY MODELS?

Figure 9 shows a comparison of the confusion matrix between the `join` (left) and the `join`+ROBERTA (right) models on the English_EWT test set. The `join`+ROBERTA model tends to show better results in classifying the NOUN, and the X typed tokens. The NOUN and PROPN are sometimes difficult to distinguish because of its ambiguity. The observations of the confusion matrix between the predicted and the true labels for each model indicate that applying ROBERTA outperforms the classification for PROPN. In detail, the total number of PROPN is 2076, and `join` and `join`+ROBERTA correctly predict 1908 and 1976 PROPNs, respectively. This difference improves the overall results by 0.31 percent. Our quantitative analyses suggest that this might be attributed to the pre-trained language model taking advantage of handling object names such as a person's name and object that appeared in Wikipedia and the common crawler [35]. With this assumption, we can also explain the gap between BERT and ROBERTA because more data are used to train the ROBERTA. In addition, ROBERTA makes our model robust for classifying code-switching when observing tag X (unclassified POS labels, typically for words in a foreign language). However, the number of tag X is only 139; thus, it may not significantly affect the overall performance.

3) STATISTICAL SIGNIFICANCE OF OUR SYSTEM

P-values are calculated by Udapi[7] by following the Universal Dependency organizer to investigate the significance of the method's results. As proposed in the previous work [36], the *p*-values are calculated by a paired bootstrap test between the predicted result of the proposed methods (last two lines in Table 4) and the baseline results (`LATTICE` and `join`, the first and the third lines). All proposed models using BERT and ROBERTA show less than 0.001 *p*-values compared with the baseline models; thus, we consider that the results are improved significantly.

B. DISCUSSION

We emphasize that the proposed character model and the joint learning approach can be applied to most NLP systems. Moreover, in recent years, many NLP systems require additional prepossessing tasks such as POS tagging and syntactic parsing because they may disambiguate the meaning of the word by adding its morphological and syntactic information. Our tagger can be used as a prepossessing system for many downstream tasks. As reported by [36], higher POS tagging results in bootstrapping the performance of dependency parsing, named entity recognition and question answering as an additional linguistic feature. However, two major limitations exist in our character models. First, the proposed character models are trained only in a supervised manner, which require a labeled training data set. Second, applying two different character models consumes 7–11 percent of the larger GPU

[7] https://github.com/udapi/udapi-python/blob/master/udapi/block/eval/

memories, and it depends on the number of attention heads. As a further discussion, our research can be extended to investigate the language-specific analysis of different languages because the number of distinct characters and types of characters varies depending on languages.

VII. CONCLUSION

In this article, we presented a novel POS tagging model with two different character-level components: one is based on word boundaries and the other considers the context at the sentence level. By training two individual taggers based on two different character models, we built a POS tagger that considers both locally optimized character information and globally optimized information regardless of the language type. In our study, we proposed three main innovations: (1) a multiattentive character model that enables the system to capture several aspects of subword information; (2) joint POS representations to combine the states of two taggers as a feature for the final tagger; and (3) contextual representation to capture context information from external resources (ELMo and BERT). Our method is effective and improves upon previously reported results in POS tagging. In future work, we plan to integrate our morphological tagger using unlabeled data to enhance subword information acquired from external resources.

REFERENCES

[1] K. Lim, C. Park, C. Lee, and T. Poibeau, "SEx BiST: A multi-source trainable parser with deep contextualized lexical representations," in *Proc. CoNLL Shared Task: Multilingual Parsing Raw Text Universal Dependencies*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 143–152. [Online]. Available: http://www.aclweb.org/anthology/K18-2014

[2] J. A. Botha, E. Pitler, J. Ma, A. Bakalov, A. Salcianu, D. Weiss, R. McDonald, and S. Petrov, "Natural language processing with small feed-forward networks," 2017, *arXiv:1708.00214*. [Online]. Available: http://arxiv.org/abs/1708.00214

[3] C. Alberti, D. Weiss, G. Coppola, and S. Petrov, "Improved transition-based parsing and tagging with neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1354–1359.

[4] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins, "Globally normalized transition-based neural networks," 2016, *arXiv:1603.06042*. [Online]. Available: http://arxiv.org/abs/1603.06042

[5] J. Giménez and L. Marquez, "Fast and accurate part-of-speech tagging: The SVM approach revisited," in *Recent Advances in Natural Language Processing III*. Amsterdam, The Netherlands: John Benjamins Publishing, 2004, pp. 153–162.

[6] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual Part-of-Speech tagging with bidirectional long short-term memory models and auxiliary loss," 2016, *arXiv:1604.05529*. [Online]. Available: http://arxiv.org/abs/1604.05529

[7] T. Dozat, P. Qi, and C. D. Manning, "Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task," in *Proc. CoNLL Shared Task: Multilingual Parsing Raw Text Universal Dependencies*, 2017, pp. 20–30. [Online]. Available: http://www.aclweb.org/anthology/K/K17/K17-3002.pdf

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018, *arXiv:1802.05365*. [Online]. Available: http://arxiv.org/abs/1802.05365

[9] W. Che, Y. Liu, Y. Wang, B. Zheng, and T. Liu, "Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation," in *Proc. CoNLL Shared Task: Multilingual Parsing Raw Text Universal Dependencies*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 55–64. [Online]. Available: http://www.aclweb.org/anthology/K18-2005

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: http://arxiv.org/abs/1810.04805

[11] D. Kondratyuk and M. Straka, "75 languages, 1 model: Parsing universal dependencies universally," 2019, *arXiv:1904.02099*. [Online]. Available: http://arxiv.org/abs/1904.02099

[12] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Inf. Fusion*, vol. 38, pp. 43–54, Nov. 2017.

[13] B. Bohnet, R. McDonald, G. Simoes, D. Andor, E. Pitler, and J. Maynez, "Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings," 2018, *arXiv:1805.08237*. [Online]. Available: http://arxiv.org/abs/1805.08237

[14] B. Bohnet, "Very high accuracy and fast dependency parsing is not a contradiction," in *Proc. 23rd Int. Conf. Comput. Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 89–97. [Online]. Available: http://dl.acm.org/citation.cfm?id=1873781.1873792

[15] K. Lim, J. Y. Lee, J. Carbonell, and T. Poibeau, "Semi-supervised learning on meta structure: Multi-task tagging and parsing in low-resource scenarios," in *Proc. AAAI Conf.*, 2020, pp. 1–8.

[16] M. Ballesteros, C. Dyer, and N. A. Smith, "Improved transition-based parsing by modeling characters instead of words with LSTMs," 2015, *arXiv:1508.00657*. [Online]. Available: http://arxiv.org/abs/1508.00657

[17] K. Cao and M. Rei, "A joint model for word embedding and word morphology," 2016, *arXiv:1606.02601*. [Online]. Available: http://arxiv.org/abs/1606.02601

[18] W. Che, J. Guo, Y. Wang, B. Zheng, H. Zhao, Y. Liu, D. Teng, and T. Liu, "The hit-scir system for end-to-end parsing of universal dependencies," in *Proc. CoNLL Shared Task: Multilingual Parsing Raw Text Universal Dependencies*. Vancouver, BC, Canada: Association for Computational Linguistics, Aug. 2017, pp. 52–62. [Online]. Available: http://www.aclweb.org/anthology/K/K17/K17-3005.pdf

[19] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Proc. AAAI*, 2016, pp. 2741–2749.

[20] X. Yu and N. Thang Vu, "Character composition model with convolutional neural networks for dependency parsing on morphologically rich languages," 2017, *arXiv:1705.10814*. [Online]. Available: http://arxiv.org/abs/1705.10814

[21] T. Shi, F. G. Wu, X. Chen, and Y. Cheng, "Combining global models for parsing universal dependencies," in *Proc. CoNLL 2017 Shared Task: Multilingual Parsing Raw Text Universal Dependencies*. Vancouver, BC, Canada: Association for Computational Linguistics, Aug. 2017, pp. 31–39. [Online]. Available: http://www.aclweb.org/anthology/K/K17/K17-3003.pdf

[22] D. Zeman *et al.*, "CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies," in *Proceedings CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2017.

[23] 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 *Proc. CoNLL Shared Task: Multilingual Parsing Raw Text Universal Dependencies*. Brussels, Belgium: ACL, Oct. 2018, pp. 1–21. [Online]. Available: http://www.aclweb.org/anthology/K18-2001

[24] C. Alberti, D. Andor, I. Bogatyy, M. Collins, D. Gillick, L. Kong, T. Koo, J. Ma, M. Omernick, S. Petrov, C. Thanapirom, Z. Tung, and D. Weiss, "SyntaxNet models for the CoNLL 2017 shared task," 2017, *arXiv:1703.04929*. [Online]. Available: http://arxiv.org/abs/1703.04929

[25] B. Bohnet, R. McDonald, G. Simoes, D. Andor, E. Pitler, and J. Maynez, "Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings," 2018, *arXiv:1805.08237*. [Online]. Available: http://arxiv.org/abs/1805.08237

[26] Z. Lin, M. Feng, C. Nogueira dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," 2017, *arXiv:1703.03130*. [Online]. Available: http://arxiv.org/abs/1703.03130

[27] A. Kulmizev, M. de Lhoneux, J. Gontrum, E. Fano, and J. Nivre, "Deep contextualized word embeddings in transition-based and graph-based dependency parsing-a tale of two parsers revisited," 2019, *arXiv:1908.07397*. [Online]. Available: http://arxiv.org/abs/1908.07397

[28] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint many-task model: Growing a neural network for multiple NLP tasks," 2016, *arXiv:1611.01587*. [Online]. Available: http://arxiv.org/abs/1611.01587

[29] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2016, *arXiv:1607.04606*. [Online]. Available: http://arxiv.org/abs/1607.04606

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[31] D. Zeman. (2018). *Universal Dependencies 2.2–CoNLL 2018 Shared Task Development and Test Data*. LINDAT/CLARIN Digital Library Inst. Formal Appl. Linguistics, Charles University, Prague. [Online]. Available: http://hdl.handle.net/11234/1-2184. http://hdl.handle.net/11234/1-2184
[32] M. Straka, J. Hajič, and J. Straková, "UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing," in *Proc. 10th Int. Conf. Lang. Resour. Eval. (LREC)*. Portorož, Slovenia: European Language Resources Association, 2016.
[33] A. Smith, B. Bohnet, M. de Lhoneux, J. Nivre, Y. Shao, and S. Stymne, "82 treebanks, 34 models: Universal dependency parsing with multi-treebank models," in *Proc. CoNLL Shared Task: Multilingual Parsing Raw Text Universal Dependencies*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 113–123. [Online]. Available: http://www.aclweb.org/anthology/K18-2011
[34] A. Smith, M. de Lhoneux, S. Stymne, and J. Nivre, "An investigation of the interactions between pre-trained word embeddings, character models and POS tags in dependency parsing," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 2711–2720.
[35] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*. [Online]. Available: http://arxiv.org/abs/1907.11692
[36] 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 *Proc. CoNLL Shared Task: Multilingual Parsing Raw Text Universal Dependencies*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1–20.

KYUNGTAE LIM received the B.S. degree in computer science from Dankook University, Seoul, South Korea, in 2010, the M.S. degree in web science technology from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2014, and the Ph.D. degree in natural language processing from the École Normale Supérieure, Paris, France, in 2020. He was a Software Engineer at Samsung, from 2010 to 2012. He worked as a full-time Researcher with the Korea Institute of Science and Technology Information (KISTI), from 2014 to 2016. He is currently a Senior Researcher with the Department of Intelligent Computing, Korea Atomic Energy Research Institute (KAERI). Since 2018, he has been contributing to the Universal Dependency Project to build Korean and Komi-Zyrian corpora. He has participated in several NLP competitions, such as the CoNLL 2017 and 2018 shared tasks and the 2018 Extrinsic Parser Evaluation task ranking at the first position for several languages. His research interest includes natural language processing, especially in dependency parsing and visual question answering.

JUNGYEUL PARK received the Ph.D. degree in linguistics from the Université Paris Diderot (Paris VII), in 2006. He was a Lecturer with the Department of Computer Science, IUT de Lannion, Université de Rennes 1, from 2013 to 2014, and a Research Scientist at CEA Nano-INNOV, from 2015 to 2016. He was a Visiting Assistant Professor with the Department of Linguistics, The University of Arizona, from 2016 to 2017, and the Department of Linguistics, SUNY Buffalo, from 2018 to 2019. He has been an Acting/Affiliate Assistant Professor with the Department of Linguistics, University of Washington, since Fall 2019. His interest in natural language processing motivated him to pursue linguistics after initially studying computer science. His current research interests include machine learning approaches for morphology and syntax of natural language.