Contextualized Character Embedding with Multi-Sequence LSTM for Automatic Word Segmentation

Hyunyoung LEE†(a) and Seungshik KANG†(b), Members

SUMMARY Contextual information is a crucial factor in natural language processing tasks such as sequence labeling. Previous studies on contextualized embedding and word embedding have explored the context of word-level tokens in order to obtain useful features of languages. However, unlike it is the case in English, the fundamental task in East Asian languages is related to character-level tokens. In this paper, we propose a contextualized character embedding method using n-gram multi-sequences information with long short-term memory (LSTM). It is hypothesized that contextualized embeddings on multi-sequences in the task help each other deal with long-term contextual information such as the notion of spans and boundaries of segmentation. The analysis shows that the contextualized embedding of bigram character sequences encodes well the notion of spans and boundaries for word segmentation rather than that of unigram character sequences. We find out that the combination of contextualized embeddings from both unigram and bigram character sequences at output layer rather than the input layer of LSTMs improves the performance of word segmentation. The comparison showed that our proposed method outperforms the previous models.

key words: contextualized character embedding, LSTM, linear-chain CRF, word segmentation

1. Introduction

Distributed representations, typically known as word embedding, are used as input in natural language processing (NLP) models, especially one based on neural networks [1], to provide useful features for various tasks. Before using neural networks with distributed representations, most of NLP tasks departed from manually designing the features. The sparseness problem happens in feature vectors, such as one-hot encoding. Distributed representations, however, encode the complex characteristics of word use (i.e., useful semantic and syntactic knowledge) into a fixed-length dense vector [2]. Therefore, it is currently the key task to extract useful information from words and improve the models of NLP tasks such as parsing [3], text classification [4]–[6], [21], machine translation [7], [8], and named entity recognition [9]–[11].

Since distributed representations are based on words as a unit [12]–[14], the word-level model encounters the out-of-vocabulary (OOV) problem with vocabulary items that have not been registered in the dictionary. Some research has attempted to resolve this problem. For example, Bojanowski et al. [15] and Park et al. [16] take into account subword-level information as additional information to represent word embeddings. Their approach improves the performance regarding both word similarity and analogy tasks using morphologically rich languages. Utilizing character embeddings not only has proved to be an efficient alternative for representation of an OOV word [15]–[17], but also has been extended to other downstream tasks such as part-of-speech [18], named entity recognition [19], and word segmentation [20]. The character embedding is used to try to construct word vector for a number of NLP tasks such as distributed representation for word vector [34] and word sequence labelling [10], [19], [22]. The character vectors are used to extract morphological information such as the prefix or suffix of a word [10], [19], [22]. They focused on constructing a word vector from its character vectors. Although it is intuitive on utilizing internal information of words for word representation, it doesn’t take account of the context of a character surrounding of it beyond the boundary of a word. However, unlike word-based NLP tasks, since the word segmentation has been tackled as character-based sequence labeling, the contexts of the target character were used as vital component.

Neural network models are powerful and popular in handling context information on a sequence of words in NLP. In particular, sequence-to-sequence problems such as machine translation and sequence tagging have shown how embedding context information into a fixed-length vector affects the qualitative performance of translation. Drawing upon the fact that recurrent neural networks (RNNs) better represent the information of recent input into a fixed-length vector, Sutskever et al. [23] used the reversed input to resolve long-range temporal dependencies. Luong et al. [7] and Bahdanau et al. [8] used context vectors with alignment mechanism, when generating target sequence in translation task, jointly learning alignment and translation. In the case of a sequence tagging task, Ma and Hovy. [10] and Chi and Nichols. [22] used the context information of the word-level to label each word with LSTM [24].

Apart from constructing a word vector from character vectors, utilizing the context of a character has also been explored [35]–[38]. These researches focused on the local window context – characters that precede and follow the target character, typically in a window of $K$ tokens on each side. They assume that a character largely depends on its neighboring characters. The local window approach of bag-of-characters context doesn’t consider the position of a char-
acter in a sentence. Hence, we replace the bag-of-characters context with globally long dependency context at sentence level, reflecting the position of a character and the global long context relation captured by either LSTM or BiLSTM.

In this paper, in order to enrich the contextualized character vector, we propose a method of capturing character-to-character relations that are far apart and out-of-reach with small window bag-of-characters. It is a character embedding method of encoding context information in n-gram (unigram and bigram) character sequences with two LSTMs in word segmentation task. Since the focus is on modeling context information of character level on multi-sequences (up to a bigram character sequence), our embeddings are contextualized on each character sequence. Our model is robust to the OOV problem because character embeddings are only used for the word segmentation task. In addition, contextualized character embeddings in multi-sequences are applied to the word segmentation task, which is the fundamental task in East Asian languages such as Korean.

2. Related Work

Word embedding faces an issue known as the OOV problem [15], [16]. In order to make it possible to deal with words that have not been registered in the vocabulary dictionary, most of NLP models based on neural networks have used character embeddings. Representing characters into vectors means complementing word-level information with character-level information. For example, Ma and Hovy. [10] and Misawa et al. [11] take character embeddings as input to their model in order to extract character-level information. The character-level information vector is then concatenated with the word embedding for the sequence labeling task.

For word segmentation, which is one of sequence labeling tasks, Kitagawa and Komachi. [20] modeled LSTM with character embeddings, character type embeddings, and character-based n-gram embeddings as inputs for Japanese word segmentation. It additionally used the dictionary vector as input to the final output layer. There has been joint word segmentation and part-of-speech (POS) tagging approach that is aimed to improve the accuracy of both tasks. Zheng et al. [18] performed joint word segmentation and POS tagging with character embeddings for Chinese. For Korean word segmentation, Lee. [25] proposed a joint model for word segmentation and POS tagging using structural SVM. Joint models usually lead to the improvement of accuracy by exploiting POS information to help word segmentation and avoid error propagation [18]. Lee. [25] showed that joint solutions can be applied to Korean. In addition, Kim and Choi. [26] also modeled bidirectional LSTM-Conditional random fields (CRF) based on a neural network to jointly integrate Korean word segmentation and POS tagging, and then it used as inputs character embeddings with n-gram features, n-gram nouns, and the POS distribution vector.

Natural language models based on neural networks have applied context information to a variety of natural language tasks. In order to resolve the issue resulting from complex syntactic and semantic characteristics of words (e.g. word polysemy), Peters et al. [27] utilized the outputs of internal layers of bidirectional LSTM on each token to represent context information between inputs and showed that LSTM outputs capture context-dependent aspects of word meaning. Contextual representation researches [27], [28], showed that the context information boosts the performance of different downstream tasks such as sentiment classification and named entity recognition. But, this paper considers context information between n-gram characters as opposed to word-based context. In other words, unlike both Kim and Choi. [26] and Kitagawa and Komachi. [20] which used various features as input, our method focuses on contextualizing character sequences.

The previous methods found that character n-grams are more effective than single characters, in particular, Wieting et al. [39] constructed a word embedding from character n-gram embeddings. It outperformed the methods that construct a word embedding from character embeddings by using LSTM and CNN. Takase et al. [40] also showed that a n-gram character embedding is useful by extending to LSTM language model. They used character n-gram embeddings to construct a word embedding. For character sequence labeling with character n-gram embedding, we propose a contextualized character n-gram with multi-sequences at a sentence level. Our method encodes character n-gram sequences to the separately contextualized character n-gram embeddings at each time with different LSTMs or Bi-LSTMs not sharing weights, meaning that n-gram character will have different embeddings depending on its global long dependency at each n-gram character sequence.

3. Contextualized Character Embedding with Multi-Sequences

The Contextualized Character Embedding with Multi-Sequences (CCEM) of the proposed model shown in Figs. 1 and 2 is straightforward and very effective compared to conventional neural network models for word segmentation that use character embedding at the input layer [18] and a variety of other inputs at the input layer [20], [26]. In contrast to the conventional models, the proposed model focuses on context-dependent aspects between character sequences using recurrent neural networks (RNNs).

An RNN works well on a variable-length sequence by maintaining a hidden state over time. It can theoretically summarize all previous contexts up to time with hidden state; however, vanilla RNN has a problem when learning long-range temporal dependencies because of vanishing and exploding gradients [29], [30]. Alternatively, LSTM [24] addresses the problem of learning long-ranges temporal dependencies by augmenting a memory state to two memory units. Apart from the memory units, LSTM utilizes a gating mechanism to control the flow of context-dependent information with gates [24]. Therefore, LSTM succeeds
in learning long-range temporal dependencies such as translation task [23] because of the two memory units and the gating mechanism. Bearing this in mind, LSTM is used for our model to utilize long-term temporal dependencies from each character sequence for the word segmentation task.

In other words, to get context information on two sequences respectively (i.e., unigram character and bigram character sequences), two separate LSTMs are used for each sequence. This section describes two contextualized character embedding methods with multi-sequences, one with unidirectional LSTM and the other one with bidirectional LSTM.

### 3.1 Unidirectional LSTM for CCEM

A simple approach with unidirectional LSTM for sequence tagging is to map input sequences to corresponding output sequences. For example, the information learned on the previous long-range temporal dependencies using cell and hidden states [9] is used to predict labels to correspond to each character. In this case, unidirectional LSTM outputs previous context-dependent information to a fixed-length vector at each location corresponding to the input and then each output vector is used to evaluate probability distribution over labels with classification layer.

In word segmentation, sentence \( s = \{c_1, c_2, \ldots, c_T\} \) consists of a sequence of characters excluding the spacing between them. In our model, a single sentence is split into two separate sequences, one of which is a sequence of unigram characters \( uni = \{c_1, c_2, \ldots, c_T\} \) and the other one is a sequence of bigram characters \( bi = \{c_1c_2, c_2c_3, \ldots, c_Tc_{T+1}\} \).

The unigram and bigram character vectors share the same space where the dimension is \( \mathbb{R}^n \). Note that a special token, \(<UNK>\) is used for unknown n-gram characters.

Two unidirectional LSTMs with multi-sequences (unigram and bigram) are designed to extract two separate contextualized types of information from each sequence, thereby separating a sentence as two types of sequences. They independently encode previous context-dependent information into a fixed-length vector, one for the context of unigram character sequence and the other for the context of bigram character sequence. The subsequent layer is the combination layer that concatenates or adds the outputs of two unidirectional LSTM encoders that do not share weights, after which the combination is turned into a representation for the final classification layer.

\[
\begin{align*}
    & h_{\text{uni}}^t = \text{LSTM}_{\text{uni}}(x_{t'}, Uni_t) \\
    & h_{\text{bi}}^t = \text{LSTM}_{\text{bi}}(x_{t'}, Bi_t)
\end{align*}
\]

Here \( Uni_t \in \{c_1, c_2, \ldots, c_T\} \), \( Bi_t \in \{c_1c_2, c_2c_3, \ldots, c_Tc_{T+1}\} \), and \( h_{\text{uni}}^t, h_{\text{bi}}^t \in \mathbb{R}^n \) summarize the contextualized information up to time \( t \) over only a sequence of unigram characters, and then \( h_{\text{uni}}^t, h_{\text{bi}}^t \in \mathbb{R}^n \) does it for only a sequence of bigram characters. Each of hidden states \( (h_{\text{uni}}^t, h_{\text{bi}}^t) \) incorporates the previous contextualized information at time \( t \) over the sequences, respectively. On the subsequent layer, the contextualized unigram character vectors and bigram character vectors are combined.

word segmentation task experiment was carried out with two types of representations \( h_t^{\text{contextualized}} = [h_{\text{uni}}^t + h_{\text{bi}}^t] \) and \( [h_{\text{uni}}^t; h_{\text{bi}}^t] \). For the combination of contextualized vectors, when making a sequence of bigram characters, a special token \(<\text{EOS}>\) was used to make the length of unigram and bigram character sequences equal.

### 3.2 Bidirectional LSTM for CCEM

The unidirectional LSTM reads an input sequence \( x = \{x_1, x_2, \ldots, x_T\} \) from left to right. This makes the contextualized embeddings depend on the left context. Hence, in order to supplement the shortcoming of the unidirectional LSTM method, another contextualized character embedding with multi-sequences is designed additionally to include right-to-left context. Bidirectional LSTMs (Bi-LSTMs) are used to perform the context-dependent encoding from left-to-right and right-to-left simultaneously.

The proposed bidirectional LSTM method is composed of the forward LSTM, denoted as \( \text{LSTM}(\cdot) \) and the backward LSTM, denoted as \( \overrightarrow{\text{LSTM}}(\cdot) \) to encode left-to-right and right-to-left contexts respectively. The forward LSTM, which is the same to unidirectional LSTM in Sect. 3.1, evaluates a sequence of forward hidden states \( \overrightarrow{h} = [h_1, h_2, \ldots, h_T] \), which is only dependent on the left context at each time \( t \). Conversely, the backward LSTM evaluates a sequence of backward hidden states \( \overleftarrow{h} = [\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_T] \), which is only dependent on the right context. The outputs of the forward and
backward LSTM can be added or concatenated to represent the left-to-right and right-to-left context using a fixed-length vector at each time t (i.e., \( h_t = [\bar{h_t}; h_t] \) or \([\bar{h_t}; \bar{h_t}]\)). In the case of using the Bi-LSTM encoder for CCEM as two unidirectional LSTMs in Sect. 3.1, after splitting a sentence into unigram and bigram character sequences as the input of the unidirectional LSTM, two separate bidirectional LSTMs are used for unigram and bigram character sequences.

\[
\begin{align*}
    h^\text{uni}_t &= \text{BiLSTM}_\text{uni}(h^\text{uni}_{t-1}, uni_i) \\
    h^\text{bi}_t &= \text{BiLSTM}_\text{bi}(h^\text{bi}_{t-1}, bi_i)
\end{align*}
\]  

(3) (4)

Taking into account the context-dependent information on each time t from left-to-right and right-to-left simultaneously, where \( h^\text{uni}_t \in \mathbb{R}^n \) summarizes the bidirectional contextualized information up to t over only a sequence of unigram characters, and then \( h^\text{bi}_t \in \mathbb{R}^n \) also does the bidirectional contextualized information for only a sequence of bigram characters. For the final classification layer, two types of contextualized character vectors with unigram and bigram character embeddings is used as input, in the same way of unidirectional LSTM, by adding \( k^\text{contextualized}_t = [h^\text{uni}_t + h^\text{bi}_t] \) or concatenating \( k^\text{contextualized}_t = [h^\text{uni}_t, h^\text{bi}_t] \). And a special token <EOS> was also used for the length of unigram and bigram sequences for the combination layer.

3.3 Classification for Word Segmentation

The word segmentation problem is tackled as a sequence labeling task: a character at the most front of the word is labeled as a B (begin) tag; except for it, the remaining characters in the word is labeled as an I (inside) tag. The final objective of word segmentation is to estimate conditional probability \( P(y|x) \), where \( x = \{x_1, x_2, \ldots, x_T\} \) denotes input sequences and \( y = \{y_1, y_2, \ldots, y_T\} \) denotes label sequences corresponding to input sequences.

CRFs are undirected graphical models trained to maximize conditional probability, which is the log-likelihood of the desired output sequence given the input sequence [31], [32]. So, in order to take advantage of a dependent score between two successive labels (i.e., transition score) including a score for output label \( y_t \) given input \( x_t \) at \( t \)-th time step, as shown in Fig. 3, we use a linear-chain CRF which is a common special-case graph structure. It captures the neighboring dependency between the prior and next labels in output sequences, so the conditional probability of the output sequence given the input sequence is evaluated by applying Eq. (5):

\[
\begin{align*}
    p(y|x) &= \frac{\exp(\sum_{k=1}^{T} U(x_k, y_k) + \sum_{k=2}^{T-1} T(y_k, y_{k+1}))}{Z(x)}
\end{align*}
\]  

(5)

Where \( U(x_k, y_k) \) is the unary score that represents how likely \( y_k \) is given corresponding to \( x_k \), \( T(y_k, y_{k+1}) \) is the transition score that represents how likely \( y_k \) is followed by \( y_{k+1} \), and \( Z(x) \) is the normalization factor used to compute the probability distribution by evaluating the score over all possible label sequences corresponding to input sequences. Note that the feedforward layer is used without an activation function such as affine transformation to get an unary score at \( t \)-th time step for the linear-chain CRF layer and a learnable transition matrix \( T \in \mathbb{R}^{n \times n} \), where \( n \) is the number of labels for the transition score. The feedforward layer can allow for the representation of a task-specific feature in a continuous vector space by increasing the number of layers. However, the capacity of the feedforward layer is limited by setting a single layer to evaluate the unary score since our goal in this work is to evaluate the capacity of contextualized embeddings for word segmentation task.

The model is trained using an end-to-end method, which means that the weight and bias terms for the unary score and transition matrix for the transition score are updated to maximize the correct output sequences given the corresponding inputs. As the highest scoring label sequences are decoded, their inference objective is expressed by Eq. (6):

\[
\hat{y} = \text{argmax}_y P(y|x)
\]  

(6)

where \( x, y, \) and \( \hat{y} \) are contextualized character embeddings, output sequences, and model predictions respectively. Once the training is finished, label sequence is produced by finding the most likely output sequence according to Viterbi algorithm [32].

4. Experiments

We evaluated our model on the dataset for a Korean word segmentation task on the sentence level, which was recently released for the 2018 Korean NLP competition1. The baseline is a unigram character embedding model that is commonly used in word segmentation [18], [20]. The dataset contains the Korean corpus. The total dataset comprises 308,825 words and 980,908 characters and is split into training and test data. In other words, it comprises 277,718 words and 882,134 characters for training, and 31,107 words and 98,774 characters for the test. The proposed model reads the corpus line by line as the input. With respect to the vocabulary dictionary, unigram and bigram characters are extracted from a sentence in the corpus. This was performed after eliminating the white-space between words in the sentence. We have described the training setting in Sect. 4.1, the result and discussion with the ablation experiment with n-gram character sequences for capacity of our

---

1https://sites.google.com/site/koreanlp2018/task-1
contextualized character embedding with multi-sequences in Sect. 4.2, and comparison with the previous models using the joint tasks in Sect. 4.3 with a focus on the word segmentation task.

4.1 Hyper-Parameters and Training Setting

We trained the model in an end-to-end way. Table 1 shows the hyper-parameters sets in training. A single feedforward layer was used before the linear-chain CRF of the final classification layer, which made it possible to focus on the capacity of contextualized character embeddings without any performance gain from multi-layer feedforward neural networks. In order to examine performance of our model, all hyper-parameters such as character embedding and batch size are set as the Table 1. When the proposed model is trained, the hidden state size of unidirectional and bidirectional LSTM is trained on the same n-gram character embedding size. For example, if the n-gram character embedding size is 300, the corresponding hidden size of both unidirectional and bidirectional LSTMs is also 300. For CCEM, a layer of both unidirectional and bidirectional LSTM is used. After Bi-LSTM combines outputs of the forward and backward via adding or concatenating, to represent contextualized character embedding with the combination of Bi-LSTM outputs if the outputs of forward and backward LSTM are added, the outputs are added for contextualized character embedding on multi-sequences with Bi-LSTM at each time step. When training is initiated, all parameters of model were initialized from a uniform distribution [33]. All parameters in stochastic gradient descent way were updated with the learning rate of 0.001. The epoch of Table 1 shows the iteration number in model training. Since the word segmentation task is tackled as sequence labeling, a linear-chain CRF was used in order to maximize the likelihood of the optimal label sequence. In practice, however, all the parameters of the model were optimized to minimize the negative log-likelihood of the optimal sequence.

4.2 Result and Discussion

To identify long-term contextual understanding such as the notion of spans and boundaries of contextualized embeddings, ablation experiments were conducted to investigate how well each n-gram sequence conveys the notion of spans and boundaries with contextualized character embedding. Table 2 shows the results. In the experiments, encoders (LSTM, Bi-LSTM) were trained by varying two types of embedding sizes (Dim ∈ {250, 300}), n-gram character sequences (unigram character sequence, bigram character sequence, or both sequences together), and approaches in combining the outputs of LSTM (Bi-LSTMs) encoders according to the n-gram character sequence types as the input for the final classification layer. In Table 2, uni and bi indicate unigram and bigram character sequences, respectively. Additionally, uni splitting a sentence into a unigram sequence is used only as the input for a single LSTM (Bi-LSTM) while bi is used for the same purpose except that it splits a sentence into a bigram sequence. Finally, uni + bi indicates that the unigram and bigram sequences are used as the input of two separate LSTMs (Bi-LSTMs) and then each output of LSTMs (Bi-LSTMs) is combined for contextualized character n-gram embedding by adding or concatenating (i.e., add or concat).

To quantify the capability of contextualized character embedding, various metrics are used. They are precision, recall, F1 of word-level and white-space location (i.e., spacing), and character tagging accuracy. The result shows that long-term contextual features such as the notion of spans and boundaries are well encoded on bigram character sequences irrespective of the encoder LSTM being unidirectional or bidirectional. This warrants that extracting contextual information from various sequences and augmenting feature information via combining it boost the performance of word segmentation. In other words, the additional contextual information from bigram character sequence augments the notion of spans and boundaries, thereby improving the performance of word segmentation. Thus, Table 2 shows that the combination of contextual information from unigram and bigram character sequences outperforms the case when single contextual information is used irrespective of the encoder being LSTM or Bi-LSTM. Consequently, we expect the combination of contextual information from unigram and bigram character sequences amplifies the informative feature to contextualized character embedding for word segmentation.

4.3 Comparison with Other Models

As shown in Table 3, the performance of our model was compared with the previous models. Our method used two types of contextualized character n-gram embeddings: one for the unidirectional LSTM and the other one for the bidirectional LSTM. The result shows that the combination of two contextualized embeddings (uni + bi) from unigram and
bigram character sequences outperforms the case when only unigram character sequence is used as input to the models, regardless of whether the LSTM is unidirectional and bidirectional. It was also found that the bidirectional LSTM model outperforms the unidirectional one in the same type of input setting in Table 3. This confirms that the bidirectional facet is the crucial component in encoding task-specific contextual features such as the notion of spans and boundaries for word segmentation.

Lee. [25] of the previous models showed that the performance of word segmentation was improved by the useful and relevant features of the other joint task, namely POS tagging. For neural networks, joint learning on multi-tasks makes the model learn the features relevant to both words segmentation and POS tagging [2], [18]; it also helps share useful features on both tasks. The model of Kim and Choi. [26] also jointly learns the neural network model on POS tagging and word segmentation simultaneously. That model is similar to the baseline model in particular, to bidirectional LSTM with unigram character sequences but it uses various features to implement the joint tasks at input layer such as syllable (character) embedding, n-gram embedding, n-gram nouns one-hot encoding vector, and the POS distribution vector. In order to attest our model is better than using character n-gram embeddings and local window context, we compared Kim and Choi. [26], which uses them at input layer, with our method to utilize globally contextualized character n-gram embeddings at sentence context.

The bidirectional factor is an important component, but the unidirectional LSTM with unigram and bigram sequences (LSTM+uni+bi) outperforms Kim and Choi. [26] based on Bi-LSTM, as well as the baseline (Bi-LSTM+uni). From these results, we assume contextual information is a key-point than exploiting POS information in identifying the notion of spans and boundaries. By enriching contextual information without any gain of multi-layer feedforward neural networks, we can see that the performance of our models is improved and outperforms the previous models, which are [25] and [26], in Table 3. In other words, this result

### Table 2

Experimental results of CCEM on the word segmentation task

|           | Dim. | Acc. | Word Pre. | Word Rec. | Word F1 | Spacing Pre. | Spacing Rec. | Spacing F1 |
|-----------|------|------|-----------|-----------|---------|--------------|--------------|------------|
| LSTM      |      |      |           |           |         |              |              |            |
| uni       | 250  | 94.814 | 81.145 | 79.371 | 80.248 | 92.32 | 90.21 | 91.252 |
| bi        | 300  | 94.655 | 80.107 | 80.162 | 80.134 | 91.05 | 91.212 | 91.084 |
| uni+bi+add| 250  | 96.985 | 88.543 | 88.497 | 88.519 | 94.99 | 94.95 | 94.969 |
| bi+concat | 300  | 96.894 | 88.712 | 87.546 | 88.125 | 95.43 | 94.13 | 94.775 |
| uni+bi+concat | 250 | 97.187 | 89.791 | 88.758 | 89.271 | 95.86 | 94.70 | 95.276 |
|            | 300  | 97.164 | 89.119 | 89.340 | 89.229 | 95.15 | 95.39 | 95.269 |

| Bi-LSTM   |      |      |           |           |         |              |              |            |
| uni+add   | 250  | 96.346 | 86.518 | 86.356 | 86.436 | 93.99 | 93.80 | 93.894 |
| baseline  | 300  | 96.178 | 86.319 | 84.906 | 85.606 | 94.38 | 92.77 | 93.568 |
| uni+concat| 250  | 96.321 | 86.425 | 86.205 | 86.314 | 93.98 | 93.73 | 93.854 |
| baseline  | 300  | 96.204 | 86.046 | 85.762 | 85.903 | 93.82 | 93.50 | 93.659 |
| bi+add    | 250  | 97.083 | 89.315 | 88.221 | 88.764 | 95.72 | 94.48 | 95.095 |
| baseline  | 300  | 97.075 | 89.458 | 87.597 | 88.517 | 96.12 | 94.03 | 95.063 |
| bi+concat | 250  | 97.042 | 88.789 | 88.552 | 88.670 | 95.19 | 94.93 | 95.059 |
| baseline  | 300  | 96.950 | 88.468 | 88.613 | 88.540 | 94.83 | 95.00 | 94.914 |
| uni+bi+add| 250  | 97.332 | **90.032** | 89.526 | **89.778** | 95.81 | 95.25 | 95.529 |
| baseline  | 300  | 97.337 | 90.030 | 89.410 | 89.718 | **95.88** | 95.19 | **95.333** |
| uni+bi+concat | 250 | 97.265 | 89.698 | **89.767** | 89.732 | 95.40 | **95.48** | 95.439 |

|            | 300  | 97.274 | 89.725 | 89.417 | 89.570 | 95.61 | 95.27 | 95.439 |

### Table 3

Comparison with other methods

| Model                  | Acc. |
|------------------------|------|
| Lee. 2013              |      |
| Structural SVM         | 96.68|
| Structural SVM + joint model | 96.86|
| Kim and Choi 2018      |      |
| Bi-LSTM-CRF            | 97.04|
| Our model              |      |
| LSTM + uni (baseline)  | 94.814|
| LSTM + uni + bi        | 97.273|
| BiLSTM + uni (baseline) | 96.346|
| BiLSTM + uni + bi      | 97.337|
indicates that the combination of the output layer of LSTMs (Bi-LSTMs) in encoding contextual n-gram character information captured by multi-sequences is a crucial component for improvement of word segmentation. It denotes that using global context information is better than character n-gram with local window context as the input layer such as Kim and Choi. [26].

5. Conclusion

We explored a contextualized character-level embedding in word segmentation, which is a fundamental task in East Asian languages. Since an input sentence can be split into a variety of sequences, it is hypothesized that context-dependent information can be extracted from both unigram and bigram character sequences. We extracted two global context information from two different LSTMs or Bi-LSTMs not sharing weights and then combined globally contextualized unigram and bigram character embeddings. Our CCEM shows even using globally contextualized bigram character embeddings at sentence context shows the better performance than using character n-gram embeddings in local window context at input layer. The combination of contextual embedding outperforms the previous models. In addition, the ablation experiment of unigram and bigram character sequences shows that the bigram character sequence encodes the more informative features to improve word segmentation rather than features from a unigram character sequence regardless of whether the LSTM is unidirectional or bidirectional. It was found that since LSTM (Bi-LSTM) encode task-specific information into a fixed-length vector at the output layer, the combination of outputs of LSTMs (Bi-LSTMs) from both unigram and bigram character sequences amplifies the useful features for word segmentation while leading to an improvement in accuracy. It was found that for word segmentation, the combination of features at the output layer of LSTM (Bi-LSTM) encodes the more informative features rather than the combination at input layer. Finally, using contextualized information to improve the word segmentation task was found to be more crucial than the joint approach.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, which helped to improve the paper. This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2017M3C4A7068186).

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” Journal of Machine Learning Research, vol.3, pp.1137–1155, 2003.
[2] R. Collobert and J. Weston, “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning,” Proc. 25th International Conference on Machine Learning, pp.160–167, 2008.
[3] D. Chen and C.D. Manning, “A Fast and Accurate Dependency Parser using Neural Networks,” Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp.740–750, Doha, Qatar, Oct. 2014.
[4] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical Attention Networks for Document Classification,” Proc. NAACL-HLT 2016, pp.1480–1489, San Diego, California, June 2016.
[5] J. Howard and S. Ruder, “Universal Language Model Fine-tuning for Text Classification,” Proc. 56th Annual Meeting of the Association for Computational Linguistics, pp.328–339, Melbourne, Australia, July 2018.
[6] M. Huang, Y. Cao, and C. Dong, “Modeling rich contexts for sentiment classification with LSTM,” arXiv preprint arXiv:1605.01478, 2016.
[7] M.-T. Luong, H. Pham, and C.D. Manning, “Effective Approaches to Attention-based Neural Machine Translation,” Proc. 2015 Conference on Empirical Methods in Natural Language Processing, pp.1412–1421, Lisbon, Portugal, Sept. 2015.
[8] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014.
[9] Z. Huang, W. Xu, and K. Yu, “Bidirectional LSTM-CRF models for sequence tagging,” arXiv preprint arXiv:1508.01991, 2015.
[10] X. Ma and E. Hovy, “End-to-end Sequence Labeling via Bi-directional LSTM-CNN-CRF,” Proc. 54th Annual Meeting of the Association for Computational Linguistics, pp.1064–1074, Berlin, Germany, Aug. 2016.
[11] S. Misawa, M. Taniguchi, Y. Miura, and T. Ohkuma, “Character-based Bidirectional LSTM-CRF with Words and Characters for Japanese Named Entity Recognition,” Proc. First Workshop on Sub-word and Character Level Models in NLP, pp.97–102, 2017.
[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” arXiv preprint arXiv:1301.3781, 2013.
[13] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” Advances in Neural Information Processing Systems, pp.3111–3119, Harrah’s and Harveys, Lake Tahoe, USA, Dec. 2013.
[14] J. Pennington, R. Socher, and C.D. Manning, “GloVe: Global Vectors for Word Representation,” Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp.1532–1543, Doha, Qatar, Oct. 2014.
[15] 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.
[16] S. Park, J. Byun, S. Baek, Y. Cho, and A. Oh, “Subword-level Word Vector Representations for Korean,” Proc. 56th Annual Meeting of the Association for Computational Linguistics, pp.2429–2438, Melbourne, Australia, July 2018.
[17] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” NIPS’15 Proc. 28th International Conference on Neural Information Processing Systems, vol.1, pp.649–657, Montreal, Canada, Dec. 2015.
[18] X. Zheng, H. Chen, and T. Xu, “Deep learning for Chinese word segmentation and POS tagging,” Proc. 2013 Conference on Empirical Methods in Natural Language Processing, pp.647–657, Seattle, Washington, USA, Oct. 2013.
[19] C. dos Santos and V. Guimarães, “Boosting Named Entity Recognition with Neural Character Embeddings,” Proc. Fifth Named Entity Workshop, joint with 53rd ACL and the 7th IJCNLP, pp.25–33, Beijing, China, July 2015.
[20] Y. Kitagawa and M. Komachi, “Long short-term memory for Japanese word segmentation,” arXiv preprint arXiv:1709.08011, 2017.
