Incorporating Word Attention into Character-Based Word Segmentation

Shohei Higashiyama[1], Masao Utiyama[1], Eiichiro Sumita[1], Masao Ideuchi[2], Yoshiaki Oida[2], Yohei Sakamoto[2], Isaac Okada[2]

[1]National Institute of Information and Communications Technology, Kyoto, Japan

[2]FUJITSU LIMITED, Tokyo, Japan

{shohei.higashiyama, mutiyama, eiichiro.sumita}@nict.go.jp, {ideuchi.masao,oida.yoshiaki,yohei.sakamoto,isaac-okada}@jp.fujitsu.com

Abstract

Neural network models have been actively applied to word segmentation, especially Chinese, because of the ability to minimize the effort in feature engineering. Typical segmentation models are categorized as character-based, for conducting exact inference, or word-based, for utilizing word-level information. We propose a character-based model utilizing word information to leverage the advantages of both types of models. Our model learns the importance of multiple candidate words for a character on the basis of an attention mechanism, and makes use of it for segmentation decisions. The experimental results show that our model achieves better performance than the state-of-the-art models on both Japanese and Chinese benchmark datasets.[1]

1 Introduction

Word segmentation is the first step of natural language processing (NLP) for most East Asian languages, such as Japanese and Chinese. In recent years, neural network models have been widely applied to word segmentation, especially Chinese, because of their ability to minimize the effort in feature engineering. These models are categorized as character-based or word-based. Word-based models (Zhang et al., 2016; Cai and Zhao, 2016; Cai et al., 2017; Yang et al., 2017) directly segment a character sequence into words and can easily achieve the benefits of word-level information. However, these models cannot usually conduct exact inference because of strategies, such as beam-search decoding and constraints of maximum word length, which are necessary as the number of candidate segmentations increases exponentially with the sentence length. On the other hand, character-based models (Zheng et al., 2013; Mansur et al., 2013; Pei et al., 2014; Chen et al., 2015a) treat word segmentation as sequence labeling. These models typically predict optimal label sequences while considering adjacent labels.

Limited efforts have been devoted to leveraging the advantages of both types of models, such as utilizing word information and conducting exact inference, which are complementary characteristics. In particular, the candidate word information for a character is beneficial to disambiguate word boundaries because a character in the sentence has multiple candidate words that contain the character. For example, there are three or four candidate words for characters x_3, x_4 and x_5 in a sentence $x_{1:5}$ in Figure 1. A feasible solution to develop a model with both characteristics is to incorporate word information into a character-based framework. An example of such work is that of Wang and Xu (2017). They concatenated embeddings of a character and candidate words and used it in their convolutional neural network (CNN)-based model. They treated candidate words equivalently, although the plausibility of a candidate word differs in the context of a target character.

In this paper, we propose a character-based word segmentation model that utilizes word information. Our model is based on a BiLSTM-CRF architecture that has been successfully applied to sequence labeling tasks (Huang et al., 2015; Chen et al., 2015b). Differing from the work of Wang and Xu (2017), our model learns and distinguishes the importance of candidate words for a character in a context, by applying an attention mechanism (Bahdanau et al., 2015).

Our contributions are as follows:

- We introduce word information and an attention mechanism into a character-based word segmentation framework, to distinguish and leverage the importance of candidate words

[1] We have released our code at https://github.com/shigashiyama/seikanlp.

Sentence:

$x_{1:5}$ = 彼は日本人 ⟨*kare wa nihonjin*⟩ (He is a Japanese.)

Candidate words $\{w_j\}$ for characters $\{x_i\}$ in the sentence:

	j	1	2	3	4	5	6	7	8	No. of
		彼	は	日	本	人	日本	本人	日本人	candidate
i	$x_i \diagdown w_j$	⟨*kare*⟩	⟨*wa*⟩	⟨*hi*⟩	⟨*hon*⟩	⟨*hito*⟩	⟨*nihon*⟩	⟨*honnin*⟩	⟨*nihonjin*⟩	words
		(he)	(NOM)	(day)	(book)	(person)	(Japan)	(the person)	(Japanese)	
1	彼	1	0	0	0	0	0	0	0	1
2	は	0	1	0	0	0	0	0	0	1
3	日	0	0	1	0	0	1	0	1	3
4	本	0	0	0	1	0	1	1	1	4
5	人	0	0	0	0	1	0	1	1	3

Figure 1: An example of candidate words $w_{1:8}$ retrieved from a vocabulary for a sentence $x_{1:5}$. Strings in angle brackets "⟨⟩" and in parentheses "()" respectively indicate (typical) readings and English translations of words. The value in each (i, j) represents whether the i-th character is contained by the j-th word (i.e., δ_{ij} in Eq. (4)).

in different contexts.

- We empirically reveal that accurate attention to proper candidate words leads to correct segmentations.
- Our model outperforms the state-of-the-art word segmentation models on both Japanese and Chinese datasets.

2 Task Definition

Word segmentation can be regarded as a character-level sequence labeling task. Given a sentence $x = x_{1:n} := (x_1, \dots, x_n)$ of length n, each character x_i will be assigned a segmentation label y_i of tag set $\mathcal{T}$, and a label sequence $y = y_{1:n}$ will be predicted. We employ tag set $\mathcal{T} = \{\mathrm{B}, \mathrm{I}, \mathrm{E}, \mathrm{S}\}$, where B, I and E, respectively, represent the beginning, inside and end of a multi-character word, and S represents a single character word (Xue, 2003).

3 Baseline Model

We use a BiLSTM-CRF architecture for our baseline model. The model consists of a character embedding layer, recurrent layers based on long short-term memory (LSTM) and a conditional random fields (CRF) layer as in Figure 2.

Character Embedding Layer Let $\mathcal{V}_c$ be a character vocabulary. Each character in a given sentence is transformed into a character embedding $\mathbf{e}^c$ of d_c-dimensional vector by a lookup operation that retrieves the corresponding column of the embedding matrix $E_c \in \mathbb{R}^{d_c \times |\mathcal{V}_c|}$.

Recurrent Layers for Character Representation A sequence of character embeddings $\mathbf{e}^c_{1:n}$ is fed into a recurrent neural network (RNN) to derive contextualized representations $\mathbf{h}_{1:n}$, which

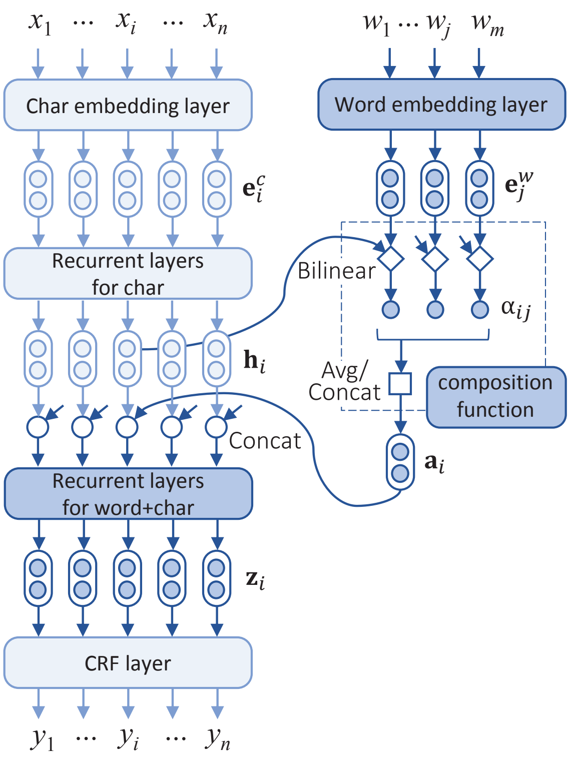

Figure 2: Architecture of our proposed model, which comprises the common components to the baseline model (light blue) and additional ones (dark blue).

we call *character context vectors*. We adopt a stacked (multi-layer) and bidirectional variant of an LSTM (Hochreiter and Schmidhuber, 1997) network, which addresses the issue of learning long-term dependencies and the gradient vanishing problem.

Hidden vectors $\mathbf{h}^{(l)}_{1:n}$ of l-th bidirectional LSTM (BiLSTM) layer are calculated by forward LSTM

(LSTM_f) and backward LSTM (LSTM_b):

$$\begin{aligned}\mathbf{h}_i^{(l)} &= \mathrm{BiLSTM}(\mathbf{h}_{1:n}^{(l-1)}, i) \qquad (1)\\ &:= \mathrm{LSTM}_f(\mathbf{h}_{1:n}^{(l-1)}, i)\\ &\quad \oplus \mathrm{LSTM}_b(\mathbf{h}_{n:1}^{(l-1)}, n-i+1)\ ,\end{aligned}$$

where $\mathbf{h}_i^{(0)} = \mathbf{e}_i^c$ and $\oplus$ denotes a concatenation operation, and $\mathbf{h}_{n:1}^{(l-1)}$ denotes the reversed sequence of the original vectors $\mathbf{h}_{1:n}^{(l-1)}$.

More concretely, each forward or backward LSTM calculates hidden vectors $\mathbf{h}_{1:n}$ from an input sequence $\mathbf{v}_{1:n}$ of d_v-dimensional vectors as follows:

$$\begin{aligned}\mathbf{h}_i &= \mathrm{LSTM}(\mathbf{v}_{1:n}, i) := \mathbf{o}_i \odot \tanh(\mathbf{c}_i)\ ,\\ \mathbf{c}_i &= \mathbf{i}_i \odot \mathbf{t}_i + \mathbf{f}_i\ \odot \mathbf{c}_{i-1},\\ \mathbf{g}_i &= \sigma(W_g \mathbf{v}_i + U_g \mathbf{h}_{i-1} + \mathbf{b}_g)\ ,\\ \mathbf{t}_i &= \tanh(W_t \mathbf{v}_i + U_t \mathbf{h}_{i-1} + \mathbf{b}_t)\ ,\end{aligned}$$

where σ is sigmoid function, $\odot$ denotes element-wise multiplication, $\mathbf{g}$ indicates an input gate $\mathbf{i}$, a forget gate $\mathbf{f}$ or an output gate $\mathbf{o}$, $W_g, U_g \in \mathbb{R}^{d_r \times d_v}$ and $\mathbf{b}_g \in \mathbb{R}^{d_r}$ are trainable parameters for each gate $g \in \{i, f, o\}$, and d_r is a hyperparameter.

CRF Layer A character context vector $\mathbf{h}_i$ is mapped into a $|\mathcal{T}|$-dimensional vector representing scores of segmentation labels:

$$\mathbf{s}_i = W_s \mathbf{h}_i + \mathbf{b}_s\ ,$$

where $W_s \in \mathbb{R}^{|\mathcal{T}| \times 2d_r}$ and $b \in \mathbb{R}^{|\mathcal{T}|}$ are trainable parameters. Following previous sequence labeling work (Collobert et al., 2011), we introduce a CRF (Lafferty et al., 2001) layer, which has a transition matrix $A \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{T}|}$ to give transition scores of adjacent labels. Thus, the score of a label sequence $y = y_{1:n}$ for a sentence $x = x_{1:n}$ is calculated as follows:

$$\mathrm{score}(x, y; \theta) = \sum_{i=1}^{n} (A_{y_{i-1}, y_i} + \mathbf{s}_i[y_i])\ ,$$

where θ denotes all the parameters and $\mathbf{s}[y]$ indicates the dimension of a vector $\mathbf{s}$ corresponding to a label y. We can find the best label sequence $y^\star$ by maximizing the sentence score:

$$y^\star = \mathrm{argmax}_{y \in \mathcal{T}^n}\ \mathrm{score}(x, y; \theta)\ . \qquad (2)$$

Training Objective During training, parameters θ of the network are learned by minimizing the negative log likelihood over all the sentences in training data $\mathcal{D}$ w.r.t θ:

$$\min_\theta \quad - \sum_{(x,y) \in \mathcal{D}} \log p(y|x, \theta)\ ,$$

$$p(y|x, \theta) = \frac{\mathrm{score}(x, y; \theta)}{\sum_{y'} \mathrm{score}(x, y'; \theta)}\ . \qquad (3)$$

Note that the Viterbi algorithm can be used for efficient calculation of the probability of a label sequence in Eq. (3) similarly to decoding in Eq. (2).

4 Proposed Model

To disambiguate word boundaries more effectively, we integrate word information into the character-based framework. More concretely, we transform embeddings of multiple candidate words for each character into a fixed size word vector, which we call a *word summary vector*, by a *word feature composition function*. We show the architecture of our model in Figure 2. In addition to the layers of the baseline model, the model comprises a word embedding layer, a word feature composition function, and additional recurrent layers.

Word Embedding Layer Given a character sequence $x = x_{1:n}$, we search for all words corresponding to subsequences of the input sequence from a word vocabulary $\mathcal{V}_w$ within a maximum word length. Then, we obtain a unique list[2] $\mathcal{W}_x$ of candidate words of size m. For example, for a given sentence $x_{1:5}$ in Figure 1, candidate words $\{w_1, \cdots, w_8\}$ will be found. Each word $w \in \mathcal{W}_x \subseteq \mathcal{V}_w$ is transformed into a d_w-dimensional vector $\mathbf{e}^w$ by the embedding matrix $E_w \in \mathbb{R}^{d_w \times |V_w|}$.

We can construct a word vocabulary to search candidate words by using external dictionaries or auto-segmented texts processed by any segmenter. Regarding the construction method used in our experiments, refer to §5.1.

Composition Functions of Word Features For a character x_i, the embeddings of all the candidate words that contain it are aggregated into a word summary vector $\mathbf{a}_i$ by a composition function. We introduce two attention-based composition functions, weighted average (WAVG) and

[2] List indicates set where each element has a unique number from 1 to the size of the set.

weighted concatenation (WCON), which enable a model to pay more or less attention according to the importance of candidate words.

Both functions calculate the importance score u_{ij} from a character x_i to a word w_j in $\mathcal{W}_x$ by a bilinear transformation, which indicates the interaction between the character context vector $\mathbf{h}_i$ and the word embedding $\mathbf{e}^w_j$. Then the weight $\alpha_{ij} \in [0, 1]$ is obtained by a softmax operation to normalize the scores:

$$\begin{aligned} u_{ij} &= \mathbf{h}_i^T W_a \mathbf{e}^w_j \,, \\ \alpha_{ij} &= \frac{\delta_{ij} \exp(u_{ij})}{\sum_{k=1}^{m} \delta_{ik} \exp(u_{ik})} \,, \end{aligned} \quad (4)$$

where $W_a \in \mathbb{R}^{2d_r \times d_w}$ is a trainable parameter. For simplification of equations, we introduce an indicator variable $\delta_{ij} \in \{0, 1\}$ that indicates whether the character x_i is included in the word w_j as Figure 1 illustrates.

Next, WAVG and WCON calculate a word summary vector $\mathbf{a}_i$ as the weighted average and the weighted concatenation of word embeddings, respectively:

$$\mathbf{a}_i = \mathrm{WAVG}(x_i, \{w_j\}_{j=1}^m) = \sum_{j=1}^{m} \alpha_{ij} \mathbf{e}^w_j \,, \quad (5)$$

$$\mathbf{a}_i = \mathrm{WCON}(x_i, \{w_j\}_{j=1}^m) = \bigoplus_{l=1}^{L} \alpha_{i,i_l} \mathbf{e}^w_{i_l} \,, \quad (6)$$

where $\{w_j\} = \mathcal{W}_x$ and $\bigoplus(\cdot)$ indicates concatenation of given arguments. Let K be the maximum word length, $L = \sum_{k=1}^{K} k$, and i_l for the character x_i denotes the corresponding index in $\mathcal{W}_x$ of l-th words w'_l in the list $\{w'_1, \ldots, w'_L\} = \bigcup_{k=1}^{K} \bigcup_{p=-k+1}^{0} \{x_{i+p:i+p+k-1}\}$. If $w'_l \notin \mathcal{V}_w$, we use a zero vector as the l-th argument in Eq. (6). For example, if $K = 3$, WCON concatenates embeddings of words corresponding to x_i (length 1), $x_{i-1:i}$, $x_{i:i+1}$ (length 2), $x_{i-2:i}$, $x_{i-1:i+1}$ and $x_{i:i+2}$ (length 3) in this order, into a single vector for the character x_i. WAVG and WCON finally output a summary vector of size d_w and Ld_w, respectively. Note that we use zero vector as a summary vector if no candidate words are found for a character.

We also use two more variants of composition functions without the attention mechanism, the average function (AVG) and the concatenation function (CON). AVG is a special case of WAVG, where $\alpha_{ij} = \delta_{ij} / \sum_k \delta_{ik}$ for all (i, j) in Eq. (5). CON is the equivalent function to the word features used in Wang and Xu (2017) and a special case of WCON, where $\alpha_{i,i_l} = 1$ for all (i, i_l) in Eq. (6).

Recurrent Layers for Word-Integrated Character Representation The summary vector $\mathbf{a}_i$ and the context vector $\mathbf{h}_i$ for a character are together fed into additional recurrent layers, which are BiLSTM layers, to further contextualize character representations using word information of surrounding characters. Given the input $\mathbf{h}_i \oplus \mathbf{a}_i$, BiLSTMs calculate hidden vectors, and the hidden vectors $\mathbf{z}_{1:n}$ of the last BiLSTM layer are fed into the CRF layer.

5 Experiments

5.1 Settings

Datasets We evaluated our model on three datasets, CTB6 and MSR for Chinese word segmentation and BCCWJ (short unit word annotation) for Japanese word segmentation. CTB6 is Chinese Penn Treebank 6.0 (Xue et al., 2005). MSR is provided by the second International Chinese Word Segmentation Bakeoff (Emerson, 2005). BCCWJ is Balanced Corpus of Contemporary Written Japanese 1.1 (Maekawa et al., 2014). We followed the same training/development/test split as in previous work (Yang and Xue, 2012; Chen et al., 2015b) for CTB6, official training/test split for MSR, and the same training/test split as in the Project Next NLP[3] for BCCWJ. We randomly selected 90% of the sentences in the training data as a training set and used the other 10% as a development set, respectively for MSR and BCCWJ.

Word Vocabulary Construction Apart from the given training and development sets for each dataset, we assumed no annotated information, including external dictionaries and third-party segmenters, was available in our experiments. Therefore, we used the training set and large unlabeled texts to obtain a word vocabulary to be used in our proposed model.

First, we trained a baseline model from each training set and applied it to large unlabeled texts. Then we collected auto-segmented words appearing in the texts[4] and gold words in the training set, and regarded the union of both kinds of words as a

[3] http://www.ar.media.kyoto-u.ac.jp/mori/research/topics/PST/NextNLP.html

[4] We discarded words occurring less than five times in auto-segmented texts, since their pre-trained embeddings were not learned by Word2Vec with the default minimum frequency of five as described later.

Parameter	Value
Character embedding size	300
Number of BiLSTM-C layers	2
Number of BiLSTM-C hidden units	600
Mini-batch size	100
Initial learning rate	1.0
Learning rate decay ratio	0.9
Gradient clipping threshold	5.0
Recurrent layer dropout rate	0.4
Word embedding size	300
Number of BiLSTM-WC layers	1
Number of BiLSTM-WC hidden units	600
Word vector dropout rate	0.2
Maximum word length	4

Table 1: Hyperparameters common between the baseline and the proposed model (top) and specific to the proposed model (bottom). BiLSTM-C and BiLSTM-WC, respectively, indicate recurrent layers for character and word-integrated character representation.

word vocabulary. We used the non-core section of BCCWJ (BCCWJ-NC)[5] for the Japanese dataset and Chinese Gigaword Fifth Edition[6] for the Chinese datasets as unlabeled texts.

Pre-training of Embedding Parameters Following previous work (Collobert et al., 2011), we pre-trained word embeddings from large texts and used them to initialize the word embedding matrix in our proposed segmenter. To pre-train word embeddings, we used the gensim (Řehůřek and Sojka, 2010) implementation of Word2Vec (Mikolov et al., 2013) and applied it to the same texts as ones used to construct the word vocabularies, i.e., the auto-segmented BCCWJ-NC or Chinese Gigaword texts processed by the baseline segmenters. We used the toolkit with a skip-gram model, embedding size 300, the number of iterations one, and other default parameters. For words occurring only in a training set, we randomly initialized their embeddings. We fine-tuned all word embeddings during training of the proposed segmenter.

In contrast, we randomly initialized all character embeddings, since pre-trained character embeddings did not improve performance in our preliminary experiments.

Hyperparameter Setting Table 1 gives the hyperparameters for the proposed model. The same dropout strategy as in Zaremba et al. (2015) was applied to non-recurrent connections of recurrent layers. We used word vector dropout, which randomly replaces a word embedding $\mathbf{e}^w$ to a zero vector when calculating a word summary vector in Eq. (5) or (6). A mini-batch stochastic gradient descent was used to optimize parameters and decayed the learning rate with a fixed decay ratio every epoch after the first five epochs. We trained models for up to 20 epochs and selected the best model on the development set.

[5] We restored provided auto-segmented texts to the original raw sentences and used them as unlabeled texts.

[6] `https://catalog.ldc.upenn.edu/ldc2011t13`

Method	BCCWJ	CTB6	MSR
BASE	98.63	95.43	96.70
AVG	98.82†	95.94†	97.63†
WAVG	98.85†	96.00†	**97.81**†‡
CON	98.84†	96.04†	97.77†
WCON	**98.93**†‡	**96.38**†‡	97.79†

Table 2: Results of the test sets. The table shows the mean of F1 scores of three runs for the baseline (BASE) and the proposed model variants. The symbols † and ‡ indicate statistical significance at 0.001 level over the baseline and over the variant without attention, respectively.

Method	BCCWJ	CTB6	MSR
Our WCON model$^\circ$	**98.9**	**96.4**	97.8
(Kitagawa and Komachi, 2018)	98.4	–	–
(Zhao and Kit, 2008)*	–	–	97.6
(Zhou et al., 2017)$^\circ$	–	96.2	97.8
(Wang and Xu, 2017)$^\circ$	–	–	**98.0**
(Liu et al., 2016)$^\circ$	–	95.5	97.6
(Zhang et al., 2016)$^\circ$	–	96.0	97.7
(Neubig et al., 2011)*	98.2*	–	–
(Sun et al., 2017)$^{\circ\bullet}$	–	96.3	97.9

Table 3: Comparison with state-of-the-art character-based (top) and word-based (middle) and other types of models (bottom) on the test sets. Models marked with a symbol indicate ones based on linear statistical algorithms ($\star$), ones with additional unlabeled texts ($\circ$) and ones replacing specific characters as preprocessing ($\bullet$). The result with $*$ is from our run on their released implementation.

5.2 Results on the Test Sets

We evaluated our baseline and proposed model variants on the test sets of the three benchmark datasets. Table 2 shows the mean of F1 scores of three runs for each dataset and each model. Among the proposed model variants, WCON achieved the best performance in almost all cases. We observed the following three findings from the results.

First, all the word-integrated model variants consistently outperformed the pure character-based baseline. We conducted McNemar's tests in a similar manner to Kudo et al. (2004), and the improvement of each variant over the base-

		Attention				Segmentation			
	Method	NoC	Lower	Upper	Acc	F1	Acc	Acc-CA	Acc-IA
BCCWJ	WAVG	2.09	30.09	94.53	79.69	99.14	99.22	99.74	**97.18**
	WCON				**90.97**	**99.21**	**99.29**	**99.88**	93.98
CTB6	WAVG	2.39	18.01	94.54	82.97	95.98	96.64	98.72	**86.50**
	WCON				**86.65**	**96.35**	**96.99**	**99.24**	82.37
MSR	WAVG	2.24	21.61	91.76	83.42	**98.53**	**98.75**	**99.65**	**94.26**
	WCON				**85.02**	98.51	98.72	99.63	93.53

Table 4: Attention accuracy, segmentation accuracy and F1-score on the development sets. The table also shows segmentation accuracy based on the cases where attention is correct (Acc-CA) and incorrect (Acc-IA), the average number of candidate words for a character (NoC) and lower and upper bounds of attention accuracy.

line was significant at 0.001 level. Second, the attention-based variants further boosted performance in comparison with their counterparts without attention. The improvements of WCON over CON on BCCWJ and CTB, and that of WAVG over AVG were statistically significant according to the McNemar's tests. We discuss the reason for the slight and insignificant performance difference between CON and WCON on MSR in §5.3. Third, the concatenation-based variants performed better than the average-based counterparts in almost all cases. This is probably because CON and WCON keep word length and character position information. For example, $(d_w + 1)$-th to $2d_w$-th dimensions of a summary vector always represent a word whose length is two and which ends with a target character (namely $x_{i-1:i}$ for x_i), while AVG and WAVG lose this kind of information.

Table 3 shows the performance of state-of-the-art models without additional annotated data. We listed only the WCON results in Table 3 since it performed the best among all variants on the development sets. In comparison with the best previous models, we obtained better performance on BCCWJ and CTB6, and achieved approximately 31% and 3% error reductions, respectively. On MSR, we obtained a comparable performance with the character-based model with word features in Wang and Xu (2017), which used different unlabeled texts from ours to pre-train word embeddings. To our knowledge, our model is the first neural network-based model that has achieved state-of-the-art results on both Japanese and Chinese word segmentation.

5.3 Analysis of Word Attention

To analyze how the attention mechanism affected segmentation performance, we show in Table 4 attention accuracy of the proposed model with the attention-based functions of the development sets. Attention accuracy regards a predicted result as correct if a character x_i most strongly attends to the word corresponding to the gold segmentation. The table also shows the segmentation performance, where accuracy indicates character-level accuracy of segmentation label prediction.

Note that the attention accuracy of a model falls between lower and upper bounds shown in the table. The upper bound indicates the ratio of characters whose candidate words contain the gold word (then attention can be correctly paid) and the lower bound indicates the ratio of characters whose candidate words consist only of the gold word (then attention is always correctly paid). For example, assuming that the gold segmentation of the sentence in Figure 1 is "$x_1|x_2|x_3x_4x_5$ $(= w_1|w_2|w_8)$", candidate words for all characters contain their gold words, and those for characters x_1 and x_2 consist only of respective gold words w_1 and w_2. Thus, the upper and lower bounds for the sentence are $5/5$ and $2/5$, respectively.

Both WAVG and WCON achieved approximately $\geq$80% attention accuracy over more than two candidate words on average. The "Acc-CA" ("Acc-IA") column denotes the segmentation accuracy in cases where the attention was correctly (incorrectly) paid. We obtained particularly high segmentation accuracy (close to or higher than 99%). However, incorrect attention led to a large drop in segmentation accuracy. Moreover, we can see a clear tendency for WCON resulting in poorer segmentation accuracy in cases with incorrect attention, compared with WAVG. This suggests that attention by WCON is more sensitive to segmentation decisions; information on attended words is more directly propagated to succeeding layers. As for the slight performance difference between CON and WCON on MSR in Table 2, a possible explanation is that existence of fewer gold words (observed from the upper bound of accuracy) leads to inaccurate attention and segmentation.

	(a)	(b)	(c)	(d)	(e)	(f)
BASE	アドレスバー	ひい | て | は	オフライン	代金 | 引換 | 金 | 額	お | 茶ノ水	昼夜
CON	アドレス | バー	ひいては	オフライン	代金 | 引換 | 金 | 額	お | 茶 | ノ | 水	昼夜
WCON	アドレス | バー	ひいては	オフ | ライン	代金 | 引換 | 金額	お | 茶 | ノ | 水	昼 | 夜
Gold	アドレス | バー	ひいては	オフ | ライン	代金 | 引換 | 金額	お茶ノ水	昼夜
	⟨*adoresu*|*bā*⟩ (address|bar)	⟨*hītewa*⟩ (besides)	⟨*ofu*|*rain*⟩ (off-|line)	⟨*daikin*|*hikikae*|*kingaku*⟩ (the amount of| payment|on delivery)	⟨*ochanomizu*⟩ (Ochanomizu)	⟨*chūya*⟩ (day and night)

Figure 3: Examples of segmentation results

w_j ＼ x_i	···	バ ⟨*ba*⟩	···	バー ⟨*bā*⟩ (bar)	ドレス ⟨*doresu*⟩ (dress)	アドレス ⟨*adoresu*⟩ (address)
ア		–		–	–	**1.00**
ド		–		–	0.21	**0.78**
レ		–		–	0.07	**0.93**
ス		–		–	0.02	**0.98**
バ		0.40		**0.52**	–	–
ー		–		**1.00**	–	–

(a)

w_j ＼ x_i	ひ ⟨*hi*⟩	い ⟨*i*⟩	て ⟨*te*⟩	は ⟨*wa*⟩	···	ひいては ⟨*hītewa*⟩ (besides)
ひ	0.00	–	–	–		**1.00**
い	–	0.00	–	–		**1.00**
て	–	–	0.00	–		**1.00**
は	–	–	–	0.00		**1.00**

(b)

w_j ＼ x_i	オ ⟨*o*⟩	フ ⟨*fu*⟩	ラ ⟨*ra*⟩	···	オフ ⟨*ofu*⟩ (off-)	ライン ⟨*rain*⟩ (line)
オ	0.00	–	–		**1.00**	–
フ	–	0.00	–		**1.00**	–
ラ	–	–	0.00		–	**1.00**
イ	–	–	–		–	**1.00**
ン	–	–	–		–	**1.00**

(c)

w_j ＼ x_i	···	代金 ⟨*daikin*⟩ (pay-ment)	引換 ⟨*hikikae*⟩ (on deli-very)	換金 ⟨*kankin*⟩ (cash-ing)	金額 ⟨*kingaku*⟩ (amount of money)
代		**1.00**	–	–	–
金		**1.00**	–	–	–
引		–	**1.00**	–	–
換		–	0	**1.00**	–
金		–	–	**0.97**	0.03
額		–	–	–	**1.00**

(d)

w_j ＼ x_i	お ⟨*o*⟩	茶 ⟨*cha*⟩ (tea)	ノ ⟨*no*⟩	水 ⟨*mizu*⟩ (water)
お	**1.00**	–	–	–
茶	–	**1.00**	–	–
ノ	–	–	**1.00**	–
水	–	–	–	**1.00**

(e)

w_j ＼ x_i	昼 ⟨*hiru*⟩ (daytime)	夜 ⟨*yoru*⟩ (night)	昼夜 ⟨*chūya*⟩ (day and night)
昼	**0.96**	–	0.04
夜	–	**0.99**	0.01

(f)

Figure 4: Weight α_{ij}, which indicates the importance from character x_i to word w_i, learned by WCON for sentences (a)-(f) in Figure 3. Weights to gold words are highlighted with blue.

5.4 Segmentation Examples

To examine segmentation results of actual sentences by different methods, we picked up sentence segments (a)-(f) from the BCCWJ development set. We show in Figure 3 the results obtained by BASE, WCON, and CON, which are selected as the character-based baseline, the best of our model variants, and its counterpart without attention, respectively. In addition, we also show values of weight α_{ij} learned by WCON in Figure 4.

In examples (a) and (b), BASE resulted in a wrong segmentation. However, both word-integrated methods correctly segmented words with the benefit of word information corresponding to gold segmentations (*adoresu*, *bar* and *hītewa*). This suggests that word information enables a model to utilize information on distant characters with target characters directly. From WCON results, we confirmed that all characters strongly attended to correct words, as in Figure 4. This suggests that accurate attention contributed to predicting correct segmentations.

In examples (c) and (d), only WCON predicted correct segmentations. The existence of correct words in the vocabulary and correct attention probably resulted in the correct segmentation for (c). As for (d), although parts of characters attended to a wrong word (*kankin*), correct attention regarding surrounding characters (*hikikae* and *kingaku*) seems to lead to the correct segmentation.

In examples (e) and (f), WCON predicted the wrong results. The wrong results of (e) by CON and WCON are probably due to the non-existence of the gold word *ochanomizu*, which is a location name, in the vocabulary. As for (f), WCON paid incorrect attention and predicted the wrong segmentation, even though the correct word *chūya* exists in the vocabulary. The model learned the incorrect weights likely due to the infrequent occurrence of the correct words; the single words *hiru* and *yoru* occur in the training set tens or hundreds of times while the compound word *chūya* occurs only twice. We may reduce these errors due to no or infrequent occurrences of gold words by increasing word vocabulary size, e.g., using larger texts to pre-train word embeddings.

6 Related Work

Word Segmentation For both Chinese and Japanese, word segmentation has been traditionally addressed by applying linear statistical algorithms, such as maximum entropy (Xue, 2003), CRF (Peng et al., 2004; Kudo et al., 2004; Zhao and Kit, 2008), and logistic regression (Neubig et al., 2011).

Various neural network architectures have been explored for Chinese word segmentation to reduce the burden of manual feature engineering. Specifically, character-based neural models have been developed to model the task as a sequence labeling problem, starting with earlier work by (Zheng et al., 2013) and (Mansur et al., 2013), which applied feed-forward neural networks. Pei et al. (2014) used a neural tensor network to capture interactions between tags and characters. More sophisticated architectures have also been used as standard components of word segmentation models to derive effective features automatically. Chen et al. (2015a) proposed gated recursive neural networks to model complicated combinations of characters. Chen et al. (2015b) used LSTM to capture long distance dependencies. Xu and Sun (2016) combined LSTM and GRNN to capture long term information better by utilizing chain and tree structures. CNNs have been used to extract complex features such as character n-grams (Chen et al., 2017) and graphical features of Chinese characters (Shao et al., 2017).

On the other hand, word-based neural models have also been proposed. Typical word-based models (Zhang et al., 2016; Cai and Zhao, 2016; Cai et al., 2017; Yang et al., 2017) sequentially determine whether or not to segment each character on the basis of word-level features and segmentation history, while keeping multiple segmentation candidates by beam search decoding. Liu et al. (2016) combined neural architectures for segment (i.e., word) representations into a semi-CRF framework, which searches for an optimal segmentation sequence consisting of variable length segments. Sun et al. (2017) proposed a gap-based model to predict whether or not to segment two consecutive characters, using a deep CNN consisting of more than ten layers.

Recent works utilized word information on a character-based framework. Zhou et al. (2017) pre-trained character embeddings using word boundary information from auto-segmented texts. Wang and Xu (2017) explicitly introduced word information into their CNN-based model. They concatenated embeddings of a character and multiple words corresponding to n-grams (n ranging from 1 to 4) that include the target character.

For Japanese, less work employed neural models for word segmentation than for Chinese. Morita et al. (2015) integrated an RNN language model into a statistical Japanese morphological analysis framework, which simultaneously segments a sentence into words and predicts word features, such as POS and lemma. Kitagawa and Komachi (2018) applied a pure neural model based on LSTM and achieved a better performance than a popular statistical Japanese segmenter (Neubig et al., 2011).

Around the same time as our work, two other character-based models for word segmentation have been proposed. Ma et al. (2018) showed a standard BiLSTM model can achieve state-of-the-art results when combined with deep learning best practices, including dropout to recurrent connections (Gal and Ghahramani, 2016) and pre-trained embeddings of character bigrams. These techniques can also be applied to and can further boost performance of our model. Yang et al. (2018) proposed a lattice LSTM-based model with subsequence (word or subword) information. Their model also considers the importance of multiple words by integrating character and word information into an LSTM cell vector using a gate-mechanism. However, their model might not fully exploit word information, since word information is given to only the first and last characters of the word.

LSTM-CRF LSTM-CRF is a popular neural architecture, which has been applied to various tagging tasks, including word segmentation (Chen et al., 2015b), POS tagging and NER (Huang et al., 2015; Ma and Hovy, 2016; Rei et al., 2016). Ma and Hovy (2016) and Rei et al. (2016) introduced the internal character information of words on word-level labeling tasks in contrast to our work introducing candidate word information of characters in the character-level labeling task.

Attention Mechanism An attention mechanism (Bahdanau et al., 2015) was first introduced in machine translation to focus on appropriate parts of a source sentence during decoding. This mechanism has been widely applied to various NLP tasks, including question answering (Sukhbaatar et al., 2015), constituency parsing (Vinyals et al., 2015), relation extraction (Lin et al., 2016) and natural language inference (Parikh et al., 2016). Rei et al. (2016) introduced a gate-like attention mechanism on their word-based sequence labeling model to determine the importance between the word itself and the internal characters for each word.

7 Conclusion and Future Work

In this paper, we proposed a word segmentation model that integrates word-level information into a character-based framework, aiming to take the advantages of both character- and word-based models. The experimental results show that our model with an attention-based composition function outperforms the state-of-the-art models on both Japanese and Chinese benchmark datasets.

Our analysis suggests that a word vocabulary with larger coverage can reduce errors deriving from unknown words. In future work, we will explore (1) the relationship between vocabulary coverage and segmentation performance, and (2) the effect of using pre-trained word vectors learned from different domain texts in domain adaptation scenarios.

Acknowledgments

We would like to thank Atsushi Fujita, Rui Wang, and the anonymous reviewers for their helpful feedback on this work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations*.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–420, Berlin, Germany. Association for Computational Linguistics.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 608–615, Vancouver, Canada. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2017. A feature-enriched neural model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3960–3966.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for Chinese word segmentation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1744–1753, Beijing, China. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for Chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206, Lisbon, Portugal. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Thomas Emerson. 2005. The 2nd international Chinese word segmentation bakeoff. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *Computing Research Repository*, arXiv:1508.01991.

Yoshiaki Kitagawa and Mamoru Komachi. 2018. Long short-term memory for Japanese word segmentation. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2880–2886.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese word segmentation with Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, Brussels, Belgium. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Kikuo Maekawa, Makoto Yamazaki, Toshinobu Ogiso, Takehiko Maruyama, Hideki Ogura, Wakako Kashino, Hanae Koiso, Masaya Yamaguchi, Makiro Tanaka, and Yasuharu Den. 2014. Balanced corpus of contemporary written Japanese. *Language Resources and Evaluation*, 48(2):345–371.

Mairgup Mansur, Wenzhe Pei, and Baobao Chang. 2013. Feature-based neural language model and Chinese word segmentation. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 1271–1277, Nagoya, Japan. Asian Federation of Natural Language Processing.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations*.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological analysis for unsegmented languages using recurrent neural network language model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2292–2297, Lisbon, Portugal. Association for Computational Linguistics.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533, Portland, Oregon, USA. Association for Computational Linguistics.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for Chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 293–303, Baltimore, Maryland. Association for Computational Linguistics.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 562–568, Geneva, Switzerland. COLING.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Marek Rei, Gamal K.O. Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318, Osaka, Japan. The COLING 2016 Organizing Committee.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 173–183, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Zhiqing Sun, Gehui Shen, and Zhihong Deng. 2017. A gap-based framework for Chinese word segmentation via very deep convolutional networks. *Computing Research Repository*, arXiv:1712.09509.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781.

Chunqi Wang and Bo Xu. 2017. Convolutional neural network with word embeddings for Chinese word segmentation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 163–172, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Jingjing Xu and Xu Sun. 2016. Dependency-based gated recursive neural network for Chinese word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–572, Berlin, Germany. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, pages 29–48.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, Vancouver, Canada. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Shuailong Liang. 2018. Subword encoding in lattice LSTM for Chinese word segmentation. *Computing Research Repository*, arXiv:1810.12594.

Yaqin Yang and Nianwen Xue. 2012. Chinese comma disambiguation for discourse analysis. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 786–794, Jeju Island, Korea. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 421–431, Berlin, Germany. Association for Computational Linguistics.

Hai Zhao and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA. Association for Computational Linguistics.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2017. Word-context character embeddings for Chinese word segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766, Copenhagen, Denmark. Association for Computational Linguistics.