A Simple and Effective Neural Model for Joint Word Segmentation and POS Tagging

Meishan Zhang, Nan Yu, and Guohong Fu

***Abstract*—Joint models have shown stronger capabilities for Chinese word segmentation and POS tagging, and have received great interests in the community of Chinese natural language processing (NLP). In this paper, we follow this line of work, presenting a simple yet effective sequence-to-sequence neural model for the joint task, based on a well-defined transition system, by using long short term memory (LSTM) neural network structures. We conduct experiments on five different datasets. The results demonstrate that our proposed model is highly competitive. By using well-trained character-level embeddings, the proposed neural joint model is able to obtain the best-reported performances in the literature.**

***Index Terms*—Chinese word segmentation, POS tagging, joint model, neural networks, transition system.**

I. Introduction

Word segmentation and POS tagging have been two fundamental tasks for Chinese natural language processing (NLP) [1], [2]. As Chinese sentences do not have explicit boundaries among words, word segmentation has been a prerequisite step, and then POS tagging is performed based on segmented sentences in aligning with the models of other languages such as English. State-of-the-art approaches treat both tasks as structural learning problems, using either sequence labeling models or transition-based incremental models [3]–[8]. The former are able to exploit dynamic decoding, while the later are flexible of feature engineering.

As word segmentation and POS tagging are highly correlative tasks, joint models which perform both tasks in a single model have achieved better performances, due to their capabilities of reducing error propagation and making use of mutual interaction between the two tasks [9]–[12]. Among the work of joint word segmentation and POS tagging, Zhang et al. (2010) [13] is one of the representative methods because of its simplicity and high efficiency. It exploits a transition-based framework, being able to use both character and word-level features. By decoding incrementally to predict a sequence of transition actions, the model behaves quite similarly to the sequence-to-sequence (Seq2Seq) neural models.

Recently, neural network models have received great attentions in the NLP community [14]. On the one hand, neural models greatly simplify the learning and decoding process of a number of NLP tasks [15]–[17], by neural embeddings [18] and powerful neural network structures such as long short term memory (LSTM) [19]. On the other hand, we can better neural model performances by pretraining techniques which aim to distill knowledge from other closely-related tasks [14]. For example, we can use pretrained word embeddings by a neural language model from a large-scale raw corpus. For both Chinese word segmentation and POS tagging, a number of neural models have been suggested, and have achieved better performances than traditional statistical models [20]–[23].

Although neural network models for separate word segmentation and POS tagging have been intensively investigated, only few work has been aware of neural joint model of the two tasks. There are several exceptions [24]–[27]. Only Kurita et al. (2017) [26] and Shao et al. (2017) [27] have achieved better performances than traditional statistical models on a widely-adopted standard CTB5 dataset . The former requires heavy feature engineering, designing more than 40 atomic features to construct their neural network structure, and the later exploits a CRF-based sequence labeling framework, resulting much lower performances than the former one.

In this work, we propose a simple yet effective Seq2Seq model for joint word segmentation and POS tagging. Our work is mainly inspired by the transition system of Zhang et al. (2010) [13], transforming the joint decoding process into a sequence of action predictions. Thus we are able to directly apply the Seq2Seq style of neural learning for the joint task. We exploit character-level embeddings including unigrams and bigrams as basic inputs, following the work of Chen et al. (2015) [28], which has demonstrated the effectiveness of both embeddings. In addition, we pretrain the input embeddings in two different ways, by using a large-scale raw corpus and an auto-tagged corpus, respectively. The pretrained embeddings on auto-tagged corpus are firstly proposed by Zhou et al. (2017) [29] for word segmentation. We make an extension to adapt the method for the joint task.

We conduct experiments on five datasets, including CTB5, CTB6, CTB7, PKU and NCC. Our neural joint model with external pretrained embeddings consistently performs better than the same transition-based model using traditional discrete features. Even without the external embeddings, our joint model can achieve comparable performances under this purely-supervised setting. We also compare the proposed joint model with the corresponding pipeline neural models, finding that the joint model always brings higher performances. By using the input character embeddings pretrained on large-scale auto-tagged corpus, our model is able to achieve the best-reported performances on the five datasets in the literature.

This work is supported by the National Natural Science Foundation of China (No. 61602160 and No. 61672211), Natural Science Foundation of Heilongjiang Province (China) grant F2016036, and Special business expenses in Heilongjiang Province (China) grant 2016-KYYWF-0183 *(Corresponding author: Guohong Fu.)*

M. Zhang, N. Yu and G. Fu are with the School of Computer Science and Technology, Heilongjiang University, Harbin 150080, China (e-mail: mason.zms@gmail.com; yunan.hlju@gmail.com; ghfu@hlju.edu.cn).

Step	Action	State	
		stack($\cdots w_{-2}\|t_{-2}\ \ w_{-1}\|t_{-1}$)	queue($c_0 c_1 \cdots$)
0	-	ϕ	ao yun $\cdots$
1	SEP(NR)	奥(ao)|NR	运(yun) 会(hui) $\cdots$
2	APP	奥运(ao yun)|NR	会(hui) 正(zheng) $\cdots$
3	APP	奥运会(ao yun hui)|NR	正(zheng) 式(shi) $\cdots$
4	SEP(AD)	奥运会(ao yun hui)|NR 正(zheng)|AD	式(shi) 开(kai) 幕(mu)
5	**APP**	**奥运会(ao yun hui)|NR 正式(zheng shi)|AD**	**开(kai) 幕(mu)**
6	SEP(VV)	奥运会(ao yun hui)|NR 正式(zheng shi)|AD 开(kai)|VV	幕(mu)
7	APP	奥运会(ao yun hui)|NR 正式(zheng shi)|AD 开幕(kai mu)|VV	ϕ

Table I
AN EXAMPLE OF THE TRANSITION SYSTEM FOR JOINT WORD SEGMENTATION AND POS TAGGING ("奥运会(AO YUN HUI, THE OLYMPIC GAMES)|NR 正式(ZHENG SHI, OFFICIALLY)|AD 开幕(KAI MU, OPEN)|VV"). THE BOLD LINE CORRESPONDS TO THE EXAMPLE IN FIGURE 1, WHICH WILL BE FURTHER ILLUSTRATED.

We make our code publicly available under Apache License at https://github.com/zhangmeishan/NNTranJSTagger.

II. THE PROPOSED MODEL

In this section, we describe the neural Seq2Seq model for joint Chinese word segmentation and POS tagging. First, we introduce the transition system of Zhang et al. (2010), which is used to linearize the joint decoding procedure. Then, we describe the neural structures of our Seq2Seq joint model in detail, which includes two core parts: the encoder and decoder, respectively. And finally, we present the training details of the proposed model.

A. The Transition System

The transition-based framework has been widely used in structural learning problems [21], [30], especially in a joint modeling setting [26], [31], [32], since it is concise and yet effective. It performs the decoding process step by step incrementally, based on a predefined transition system. Concretely, a transition system has two key components: (1) transition states and (2) a set of transition actions. A transition state defines representation of a partial result, while transition actions are used to control how a transition state advance by one step. Initially, we have an empty starting state, and then the state advances gradually by a sequence of transition actions, until it reaches an end state representing a full result.

By using a well-designed transition system of joint word segmentation and POS tagging, we are able to linearize the decoding process naturally into predicting a sequence of transition actions. Thus we can then directly apply a neural Seq2Seq model for the joint task. In addition, we are capable of using both character-level and word-level features during the decoding phase, by taking the accompanying transition states into account.

We follow Zhang et al. (2010) [13] to define our transition system, which has achieved state-of-the-art performances for joint word segmentation and POS tagging with traditional discrete features. The transition state consists of a stack and a queue, where the stack preserves the partially-analyzed segmentation and POS tagging results, and the queue holds the unprocessed Chinese characters. The transition system defines two kinds of actions:

- SEP(t): move the first character of the queue onto the stack as a new (sub)word with POS tag t.
- APP: move the first character of the queue onto the stack, appending it to the top-stack (sub)word.

We provide an example to illustrate the transition system, as shown in Table I. Given an input sentence "奥(ao)运(yun)会(hui)正(zheng)式(shi)开(kai)幕(mu)" (The Olympic Games officially open), we can obtain the final result "奥运会(ao yun hui, the Olympic Games)|NR 正式(zheng shi, officially)|AD 开幕(kai mu, open)|VV" by a sequence of actions "SEP(NR) APP APP SEP(AD) APP SEP(VV) APP", according to the transition system.

B. Seq2Seq Modeling

Neural Seq2Seq learning has received increasing interests in the NLP community. On the one hand, it can give state-of-the-art performances for a number of NLP tasks, including machine translation [33], dialogue [16] and question answering [17]. And on the other hand, it is simple and easy to extend. All required for a task is to convert it into predicting a sequence of symbols. Here we convert the joint word segmentation and POS tagging into a sequence of actions based on the aforementioned transition system.

A Seq2Seq model consists of two parts: (1) an encoder that represents source input sequences, and (2) a decoder that incrementally predicts the next coming symbol. The overall neural network structure of the joint model is depicted in Figure 1. In our proposed Seq2Seq model, the encoder is used to represent input Chinese character sequences, as shown by the bottom region of the figure, and the decoder is used to predict the transition action sequences, as shown by the upper region of the figure. In the following, we describe the encoder and decoder parts, respectively.

C. Encoder

In this work, as shown by the bottom part of Figure 1, we use a bi-directional LSTM [19] to encode input Chinese character sequences, following the majority Seq2Seq models [16], [33], [34]. Give a sequence of Chinese characters $c_1 c_2 \cdots c_n$, the bi-directional LSTM is built as follows. First, we derive two sequences of input features $\overrightarrow{\mathbf{x}}_1 \overrightarrow{\mathbf{x}}_2 \cdots \overrightarrow{\mathbf{x}}_n$ and $\overleftarrow{\mathbf{x}}_1 \overleftarrow{\mathbf{x}}_2 \cdots \overleftarrow{\mathbf{x}}_n$ from the discrete input character symbols by

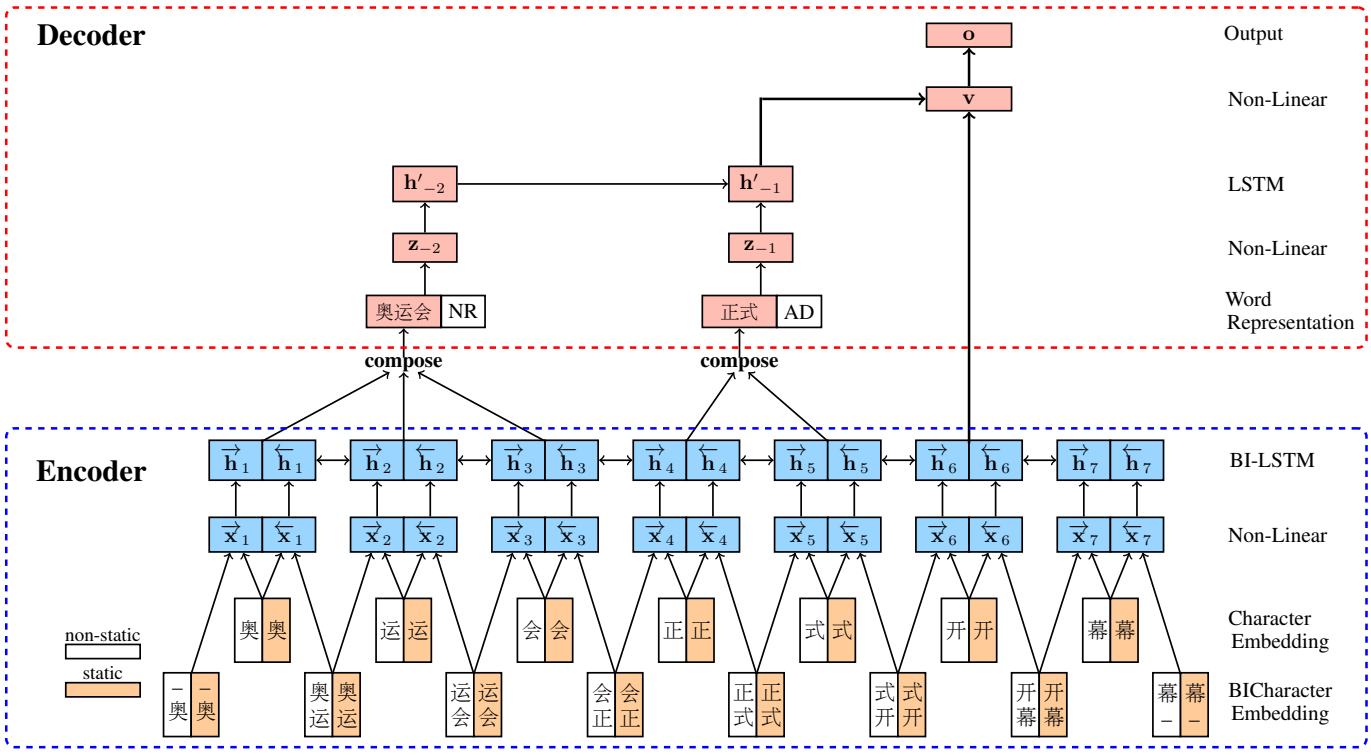


Figure 1. An example to illustrate the proposed model, which corresponds to the state after applying the fifth action APP as shown in Table I by the bold line, and is going to predict the sixth action, where "–" is a mark to denote the start or the end of a sentence.

neural embedding, resepctively. And then, we apply a left-to-right LSTM over $\overrightarrow{\mathbf{x}}_1 \overrightarrow{\mathbf{x}}_2 \cdots \overrightarrow{\mathbf{x}}_n$ and a right-to-left LSTM over $\overleftarrow{\mathbf{x}}_1 \overleftarrow{\mathbf{x}}_2 \cdots \overleftarrow{\mathbf{x}}_n$ to obtain the bi-directional hidden outputs of LSTMs, respectively.

Embedding Layer. We make embeddings from two types of discrete sources: character unigram c_t and bigram $c_t c_{t+1}$ ($t \in [1, n]$) in this work. Character bigrams have been shown highly effective for Chinese word segmentation in a number of studies, under the neural setting [21], [28].

Formally, we exploit two looking-up tables $\mathbf{E}^c$ and $\mathbf{E}^{bc}$ for character unigrams and bigrams, respectively. Give one character unigram c_t, its embedding $\mathbf{E}^c_{c_t}$ is obtained by indexing from $\mathbf{E}^c$, and given one character bigram $c_t c_{t+1}$, its embedding $\mathbf{E}^{bc}_{c_t c_{t+1}}$ is obtained by indexing from $\mathbf{E}^{bc}$ accordingly. The two looking-up tables are model parameters which would be learned during training by the back-propagation loss from the final training objective.

LSTM Inputs. In this work, the bi-directional LSTM takes different neural features as inputs due to the exploration of character bigram embeddings. In both left-to-right and right-to-left settings, the input representations at position t always use the features ending at the character c_t, where the last input character bigrams for the left-to-right and right-to-left LSTMs are $c_{t-1}c_t$ and $c_t c_{t+1}$, respectively.

In detail, we produce the input features at position t of the left-to-right LSTM only by embeddings of c_t and $c_{t-1}c_t$, while for the right-to-left LSTM, we use the embeddings of c_t and $c_t c_{t+1}$ to produce the corresponding input features at position t. We use a non-linear feed-forward neural layer to combine embeddings of character unigrams and bigrams, which can be formalized as follows:

$$\begin{aligned} \overrightarrow{\mathbf{x}}_t &= \tanh(\mathbf{W}_c[\mathbf{E}^c_{c_t}, \mathbf{E}^{bc}_{c_{t-1}c_t}] + \mathbf{b}_c) \\ \overleftarrow{\mathbf{x}}_t &= \tanh(\mathbf{W}_c[\mathbf{E}^c_{c_t}, \mathbf{E}^{bc}_{c_t c_{t+1}}] + \mathbf{b}_c), \end{aligned} \tag{1}$$

where $\mathbf{W}_c$ and $\mathbf{b}_c$ are model parameters.

Bi-Directional LSTM. After receiving the two sequences of features $\overrightarrow{\mathbf{x}}_1 \overrightarrow{\mathbf{x}}_2 \cdots \overrightarrow{\mathbf{x}}_n$ and $\overleftarrow{\mathbf{x}}_1 \overleftarrow{\mathbf{x}}_2 \cdots \overleftarrow{\mathbf{x}}_n$, we apply two LSTMs over them in a left-to-right order and a right-to-left order,respectively, obtaining the final encoder outputs $\mathbf{h}_t = \overrightarrow{\mathbf{h}}_t \oplus \overleftarrow{\mathbf{h}}_t$ ($t \in [1, n]$).

Taking the left-to-right LSTM as an example, we compute its hidden outputs $\overrightarrow{\mathbf{h}}_1 \overrightarrow{\mathbf{h}}_2 \cdots \overrightarrow{\mathbf{h}}_n$ incrementally as follows:

$$\begin{aligned} \overrightarrow{\mathbf{i}}_t &= \sigma(\overrightarrow{\mathbf{W}}_i \overrightarrow{\mathbf{x}}_t + \overrightarrow{\mathbf{U}}_i \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{V}}_i \overrightarrow{\mathbf{c}}_{t-1} + \overrightarrow{\mathbf{b}}_i) \\ \overrightarrow{\mathbf{f}}_t &= \sigma(\overrightarrow{\mathbf{W}}_f \overrightarrow{\mathbf{x}}_t + \overrightarrow{\mathbf{U}}_f \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{V}}_f \overrightarrow{\mathbf{c}}_{t-1} + \overrightarrow{\mathbf{b}}_f) \\ \overrightarrow{\mathbf{o}}_t &= \sigma(\overrightarrow{\mathbf{W}}_o \mathbf{x}_t + \overrightarrow{\mathbf{U}}_o \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{V}}_o \overrightarrow{\mathbf{c}}_t + \overrightarrow{\mathbf{b}}_o) \\ \overrightarrow{\mathbf{c}}_t &= \overrightarrow{\mathbf{f}}_t \odot \overrightarrow{\mathbf{c}}_{t-1} + \overrightarrow{\mathbf{i}}_t \odot \tanh(\overrightarrow{\mathbf{W}}_c \overrightarrow{\mathbf{x}}_t + \overrightarrow{\mathbf{U}}_c \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}_c) \\ \overrightarrow{\mathbf{h}}_t &= \overrightarrow{\mathbf{o}}_t \odot \tanh(\overrightarrow{\mathbf{c}}_t) \end{aligned} \tag{2}$$

where $\overrightarrow{\mathbf{i}}_t$, $\overrightarrow{\mathbf{f}}_t$ and $\overrightarrow{\mathbf{o}}_t$ are input, output and forget gates which aim to control the information flow inside the LSTM structure, $\overrightarrow{\mathbf{U}}$, $\overrightarrow{\mathbf{V}}$, $\overrightarrow{\mathbf{W}}$ and $\overrightarrow{\mathbf{b}}$ are model parameters, and $\odot$ denotes Hadamard product. For the right-to-left LSTM, simply reverse the computation direction based on $\overleftarrow{\mathbf{x}}_1 \overleftarrow{\mathbf{x}}_2 \cdots \overleftarrow{\mathbf{x}}_n$.

Pretrained Embeddings. Recently, a number of studies have demonstrated that pretrained embeddings are able to bring improved performances in natural language processing [14], [35], [36], including Chinese word segmentation [37],

[38]. In this work, we investigate the effectiveness of pretrained embeddings as well.

We pretrain embeddings for character unigrams and bigrams both. We follow Dyer et al. (2015) [36], augmenting the representations of character unigrams and bigrams by the external pretrained embeddings. The pretrained embeddings are kept in two additional looking-up tables $\tilde{\mathbf{E}}^c$ and $\tilde{\mathbf{E}}^{bc}$ for character unigrams and bigrams, respectively. Thus the calculation Equation (1) of the bi-directional LSTM inputs is changed into:

$$\begin{aligned}\overrightarrow{\mathbf{x}}_t &= \tanh(\mathbf{W}_c[\mathbf{E}^c_{c_t} \oplus \tilde{\mathbf{E}}^c_{c_t}, \mathbf{E}^{bc}_{c_{t-1}c_t} \oplus \tilde{\mathbf{E}}^{bc}_{c_{t-1}c_t}] + \mathbf{b}_c) \\ \overleftarrow{\mathbf{x}}_t &= \tanh(\mathbf{W}_c[\mathbf{E}^c_{c_t} \oplus \tilde{\mathbf{E}}^c_{c_t}, \mathbf{E}^{bc}_{c_t c_{t+1}} \oplus \tilde{\mathbf{E}}^{bc}_{c_t c_{t+1}}] + \mathbf{b}_c),\end{aligned} \tag{3}$$

The two tables $\tilde{\mathbf{E}}^c$ and $\tilde{\mathbf{E}}^{bc}$ are fixed and would not be fine-tuned during training, thus we are able to use the embeddings out of the training data, because the training process does not change their original distributions. We refer these kinds of fixed embeddings as **static embeddings**, while refer the fine-tuned embeddings obtained from $\mathbf{E}^c$ and $\mathbf{E}^{bc}$ as **non-static embeddings**, as shown in Figure 1 differentiated by color.

Basic Embeddings & Word-Context Embeddings. Here we propose two different methods to pretrain the embeddings of character unigrams and bigrams. The first kind of embeddings is pretrained on a raw corpus with only language model information, while the second kind of embeddings is pretrained on an auto segmented and POS tagged corpus, where the richer word-level context information such as segmentation and POS is included. We refer the two kinds of embeddings as ***basic embeddings*** and ***word-context embeddings***, respectively.

The first kind of embeddings is widely used by previous neural word segmentation models [20], [21], [24], [28], [37], [39]. The second kind of embeddings is inspired by recent studies of Yang et al. (2017) [38] and Zhou et al. (2017) [29], which pretrain word-context embeddings for neural word segmentation, where only word boundary information is included in their methods. In this work, we include word-level POS information as well to enhance the joint task.

In detail, we exploit the skip-gram model [18] to pretrain the two kinds of embeddings on large-scale corpus. The model is re-interpreted by Levy and Goldberg (2014) [40], making it capable of supporting arbitrary features. Given one input unit u (a character unigram or bigram in this work), the skip-gram model aims to predict its surrounding features f. During the training, it aims to maximize the following objective function:

$$\sum_{(u,f)\in C} \log\sigma(\tilde{\mathbf{E}}_u \cdot \tilde{\mathbf{E}}'_f) + \sum_{(u,f)\in \overline{C}} \log\sigma(-\tilde{\mathbf{E}}_u \cdot \tilde{\mathbf{E}}'_f),$$

where $\sigma(\cdot)$ denotes the sigmoid function, C denotes the set of unit-feature pairs occurring in the corpus, and $\overline{C}$ denotes those of negative pairs, $\tilde{\mathbf{E}}$ is the pretrained static embeddings which will be used in our neural joint model.

To pretrain ***basic embeddings***, we extract four surrounding features for each unit. For example, given a raw sentence $c_1c_2\cdots c_n$, we extract the four features c_{t-2}, c_{t-1}, c_{t+1} and c_{t+2} for each character unigram c_t, and extract similar four features $c_{t-2}c_{t-1}$, $c_{t-1}c_t$, $c_{t+1}c_{t+2}$ and $c_{t+2}c_{t+3}$ for each character bigram c_tc_{t+1}. All these features reflect only language model information as shown by the example.

To pretrain ***word-context embeddings***, we associate all surrounding features above with richer word context information, including word boundaries and POS tags. We achieve this goal simply by defining a special character-level label for each character that is able to represent word context information. After then, the new surrounding features are defined over the same characters augmented with well-defined character-level labels. For example, a feature c_{t+1} is changed into $c_{t+1}l_{t+1}$, and similarly a feature $c_{t+1}c_{t+2}$ is changed into $c_{t+1}l_{t+1}c_{t+2}l_{t+2}$, where l_{t+1} and l_{t+2} are corresponding character-level labels for c_{t+1} and c_{t+2}, respectively.

In order to obtain the character-level labels, we segment and tag the same raw corpus of pretraining basic embeddings by a discrete joint model ZPar [13]. Given an auto segmented and POS-tagged sentence $w_1|p_1\ w_2|p_2\ \cdots\ w_m|p_m$, we exploit the `BMES` schema to convert the word-level POS tags into character-level labels, where the beginning character of w_t is tagged as `B`-p_t, the ending character is tagged as `E`-p_t and the remaining characters of w_t are labeled as `M`-p_t. If w_t is a single-character word, we tag the character as `S`-p_t.

D. Decoder

The decoder aims to find a next-step action conditioned on historical actions. The overall decoding framework of our Seq2Seq model is shown by the upper part of Figure 1. Following typical Seq2Seq models such as [17], [34], on the one hand, we extract a source of features from encoder outputs for prediction, and on the other hand, we build a left-to-right LSTM over the generated output words incrementally, which is exploited as another source of features for action classification.

Decoder LSTM. For the left-to-right decoder LSTM, assuming the word sequence is $w_{-m}\cdots w_{-2}w_{-1}$, first we represent the discrete sequential words into dense hidden vectors $\mathbf{z}_{-m}\cdots\mathbf{z}_{-2}\mathbf{z}_{-1}$, which will be introduced later, and then we calculate the LSTM outputs $\mathbf{h'}_{-m}\cdots\mathbf{h'}_{-2}\mathbf{h'}_{-1}$ incrementally by using the same Equation (2).

Output Layer. When the decoder LSTM is ready, we are able to compute a one-step (e.g., at step t) output by the following Equation:

$$\begin{aligned}\mathbf{v} &= \tanh(\mathbf{W}_{\text{feature}}[\mathbf{h}_t, \mathbf{h'}_{-1}] + \mathbf{b}_{\text{feature}}) \\ \mathbf{o} &= \mathbf{W}_{\text{output}}\mathbf{v}\end{aligned} \tag{4}$$

where $\mathbf{h}_t$ and $\mathbf{h'}_{-1}$ are selected features from encoder and decoder LSTMs, respectively, $\mathbf{W}_{\text{feature}}$, $\mathbf{W}_{\text{output}}$ and $\mathbf{b}_{\text{feature}}$ are model parameters, and the dimension size of $\mathbf{o}$ equals the number of transition actions. The best-scored action $a_{\max} = \text{argmax}_a\, \mathbf{o}_a$ is chosen as the next step action.

Comparison with Typical Seq2Seq Models. To clearly understand the proposed Seq2Seq model for joint word segmentation and POS tagging, we make comparisons with representative Seq2Seq models such as [17], [34], showing two significant differences of our proposed model.

First, we do not require an attention mechanism, as shown in Equation (4), we use the t-th hidden vector of encoder outputs at step t, while other Seq2Seq models commonly apply

an attention neural network to align with encoder outputs in order to find important features automatically. This is because that the next-step action is defined straightforwardly over a certain input position in our transition system.

Table I shows an example to illustrate it. As shown by the bold line at step 5, the next coming (sixth) action SEP(VV) is exactly corresponding to the sixth character of the sentence (the character "开(kai)"), which is on top of the queue. The action is used to label "开(kai)" as a beginning character of the next word with POS tag VV by definition. Thus our attention is explicitly defined by the accompanying transition state.

Second, since word-level features for both segmentation and POS tagging are highly important, the decoder LSTM is built over the output word sequences instead of the sequence of predicted actions which are character level in essential. Thanks to the accompanying transition state, we are able to obtain the whole sequence of words during decoding, as well as their POS tags which could be used to enhance word representation. As shown by the stack column of Table I, we can see that the word-tag sequence is "奥运会(ao yun hui)|NR 正式(zheng shi)|AD" after applying the fifth action APP.

Word Representation. We compute vector representations $\mathbf{z}_{-m} \cdots \mathbf{z}_{-2}\mathbf{z}_{-1}$ of the produced words on stack by two folds: their POS tags and their character sequences. As shown in Figure 1, for a given word $w_{-i}, i \in [1, m]$, on the one hand, we use a neural embedding layer to convert its POS tag p_{-i} into a dense vector $\mathbf{E}^p_{p_{-i}}$, where $\mathbf{E}^p$ is a looking-up matrix for POS tags, which is a model parameter initialized randomly and would be updated during training, similar to the looking-up tables of non-static character embeddings.

On the other hand, we obtain the word's vector representation by its characters $w_{-i} = c_s \cdots c_t$ as one complementary. For computation efficiency, we directly use the encoder outputs to compose the representation. We investigate several methods, including widely-used max, min, average pooling and recently suggested self-attention pooling [41], as well as the LSTM-Minus proposed by Wang and Chang (2016) [42].

For all the pooling methods, the word representation can be computed as follows:

$$\mathbf{h}_{w_{-i}} = \sum_{k=s}^{t} a_k \mathbf{h}_k,$$

where

$$a^{\text{min}}_{k,d} = \begin{cases} 1, & \text{if } k = \text{argmin}_{j \in [s,t]} \mathbf{h}_{j,d} \\ 0, & \text{otherwise} \end{cases}$$

$$a^{\text{max}}_{k,d} = \begin{cases} 1, & \text{if } k = \text{argmax}_{j \in [s,t]} \mathbf{h}_{j,d} \\ 0, & \text{otherwise,} \end{cases}$$

$$a^{\text{average}}_k = \frac{1}{t-s+1}$$

$$a^{\text{attention}}_k \propto \exp\big(\tanh(\mathbf{W}_{\text{attention}}\mathbf{h}_k + \mathbf{b}_{\text{attention}})\big),$$

where $\mathbf{W}_{\text{attention}}$ and $\mathbf{b}_{\text{attention}}$ are two model parameters for self-attention pooling.

		CTB5	CTB6	CTB7	PKU	NCC
Train	#sent	18k	23k	31k	17k	17k
	#word	494k	641k	718k	993k	482k
Devel	#sent	350	2.1k	10k	1.9k	1.9k
	#word	6.8k	60k	237k	100k	53k
	#oov	553	3.3k	13k	3.8k	3.3k
Test	#sent	348	2.8k	10k	2.5k	3.6k
	#word	8.0k	82k	245k	153k	97k
	#oov	278	4.6k	13k	6.5k	5.1k
#POS Tag		33	33	33	40	63

Table II
STATISTICS OF DATASETS.

For the LSTM-minus approach, the word representation is computed by the subtraction of LSTM outputs. For the left-to-right LSTM, we get a word representation by formula $\overrightarrow{\mathbf{h}}_{w_{-i}} = \overrightarrow{\mathbf{h}}_t - \overrightarrow{\mathbf{h}}_{s-1}$. And similarly, the representation obtained from the right-to-left LSTM is calculated by using $\overleftarrow{\mathbf{h}}_{w_{-i}} = \overleftarrow{\mathbf{h}}_s - \overleftarrow{\mathbf{h}}_{t-1}$. Then the word representation of this part is denoted by $\mathbf{h}_{w_{-i}} = \overrightarrow{\mathbf{h}}_{w_{-i}} \oplus \overleftarrow{\mathbf{h}}_{w_{-i}}$. This approach has been used in Zhang et al. (2017) [43] to represent words in a similar transition-based model, showing great effectiveness on the Chinese word segmentation task.

Finally, we combine the above two parts by a non-linear composition, obtaining the word's final representation $\mathbf{z}_{-i}$, which is formalized as:

$$\mathbf{z}_{-i} = \tanh(\mathbf{W}_{\text{compose}}[\mathbf{h}_{w_{-i}}, \mathbf{E}^p_{p_{-i}}] + \mathbf{b}_{\text{compose}}), \quad (5)$$

where $\mathbf{W}_{\text{compose}}$ and $\mathbf{b}_{\text{compose}}$ are two model parameters.

E. Training

We train the proposed model by using a simple cross-entropy loss, following previous studies on deterministic transition-based neural models [36], [44], [45]. On the one hand, the training is highly efficient, and on the other hand, according to our preliminary experiments, we find that the cross-entropy objective has been already to give strong performances, and we are unable to obtain significantly better results with global learning proposed by Andor et al. [46] to maximize sentence-level likelihoods or F-measure scores.

For each prediction, assuming that the gold-standard transition action is a_g, and the resulting score vector for all transition actions is $\mathbf{o}$, the loss function is defined as follows:

$$\text{loss}(\Theta, a_g) = -\log p_{a_g} = -\log \frac{\exp(\mathbf{o}_{a_g})}{\sum_i \exp(\mathbf{o}_i)}, \quad (6)$$

where Θ is the set of all model parameters, p_{a_g} is the probability of the gold-standard transition action, which is computed by softmax over the output vector.

We apply online learning with a mini-batch size 16 for training, updating model parameters at the sentence level by using Adam [47] with a learning rate 10^{-3}. We exploit the dropout technique with a ratio 0.25 to avoid overfitting.

III. EXPERIMENTS

A. Experimental Settings

We use the Chinese Penn Treebank version 5.0, 6.0 and 7.0 to conduct the experiments, splitting the corpora into

training, development and test sections according to Zhang et al. (2014) [32]. In addition, we perform experiments on PKU and NCC datasets as well, where the PKU dataset is selected from People's Daily corpus, and the NCC dataset is from the POS tagging task of the Fourth International Chinese Language Processing Bakeoff [48].[1] Table II shows the corpus statistics. Although CTB5 has been widely used to evaluate the joint models of Chinese word segmentation and POS tagging, it has biases due to the small scale of the test corpus. Thus we use other four additional datasets to verify our models.

We exploit the standard word-level method to evaluate the model performances for both word segmentation and POS tagging, computing word-level precisions (P), recalls (R) and their F-measures (F), and exploiting the F-measure values as the major metrics. A word is correctly recognized when its start/end boundaries are both correct, and a POS tag is correct only when the corresponding word and its label are both correct. Thus the POS tagging performance refers to the performance of joint word segmentation and POS tagging.

The major hyper-parameters in the proposed model are dimension sizes of neural feature vectors, all of which are set to 200 in our experiments, except that the dimension size of POS tag embeddings is 50. We use *word2vecf*[2] to pretrain character-level embeddings. All the embeddings, including the basic and the word-context embeddings of character unigrams and bigrams, are pretrained on the Chinese Gigaword corpus (LDC2011T13). In order to obtain the auto segmented and POS-tagged corpus of pretraining word-context embeddings, we use ZPar[3] to train a joint model on the corresponding training corpus, and then segment and tag the raw Gigaword corpus by the trained model.

B. Baseline Systems

We compare our neural joint model with two baseline systems mainly. The first system is **ZPar** [13], presented in the above Section for pretraining of word-context character-level embeddings. The system is a transition-based model that exploits discrete manually-crafted features, which has achieved state-of-the-art performances on joint word segmentation and POS tagging among the statistical models. By comparing with this system, we verify the effectiveness of neural features proposed in this work.

In addition, we also compare our system with a state-of-the-art **pipeline** system based on comparable neural features. The pipeline system consists of two sub systems:

- **Word Segmentation**. The pipeline neural word segmentation model is implemented by removing all tag-related components of our neural joint model. In detail, the model has a same encoder compared with our neural joint model, a similar decoder without POS tag emeddings in the left-to-right word LSTM, and only two transition actions since `SEP` has no arguments. We exploit the same character-level embeddings as the neural joint model. In particular, Zhang et al. (2016) [21] have proposed a similar neural segmentation model based on the same transition system, and it enhances their model performances by using more complicated neural features such as pretrained word embeddings and beam-search decoding. We find that this simple Seq2Seq word segmentation model is able to achieve better performances compared with their model. On the same CTB6, PKU and MSR datasets adopted in Zhang et al. (2016) [21], this model obtains testing F-measure scores of 95.58%, 95.72% and 97.38%, respectively, while the corresponding numbers in their paper are 95.01%, 95.1% and 97.0%, respectively.
- **POS Tagging**. The pipeline neural POS tagging model is also a Seq2Seq model, using a word-level bi-directional LSTM as encoder together with a left-to-right incrementally constructed tag LSTM to predict the next-step POS tag sequentially, which is similar to our joint model. Essentially, it adds an incremental tag LSTM in comparison with Wang et al. (2015) [49], which demonstrates effective in our preliminary experiments for Chinese POS tagging. As character-level features have also shown highly helpful for neural POS tagging [50], we build a convolutional neural network based on the covering characters of a word to enhance word representation. For the character-level embeddings, we use the same settings as our joint model. For the word embeddings, we use the same corpus of pretraining word-context embeddings to pretrain them.

The pipeline system is closely related with our neural joint model in design for fair comparisons. The input character-level embeddings always keep the same as our neural joint model. In addition, the pipeline system can use additional word embeddings, which are necessary inputs for neural POS tagging, however these embeddings are unable to bring improved performances in our joint model by integrating them to enhance our word representation as described by Equation (5) according to our preliminary observation.

C. Model Details

The proposed neural joint model largely reduces the total number of model parameters compared with its pipeline system mentioned above, amounting a similar scale of model parameters with the baseline word segmentation, since they are only different in the output dimension of the output layer, which equals the number of transition actions. The number is below 64 in all our datasets, while the input and output dimension sizes of other neural layers are all no less than 200. The proposed joint model exploits only necessary neural structures with minimum model parameters, including char unigram and bigram embeddings, parameters of their compositions aiming for dimension reduction, parameters of a one-layer char Bi-LSTM, a left-to-right word LSTM and two feed-forward neural layers. These model parameters are all quite common in neural word segmentation models. We remove word-level embeddings which are necessary model parameters for the pipeline neural POS tagging model.

[1] We are unable to obtain the annotated PKU testing corpus of Jin and Chen (2008)[48], thus directly use the annotated data chosen from People's Daily corpus instead.

[2] https://bitbucket.org/yoavgo/word2vecf/

[3] https://github.com/SUTDNLP/ZPar

Model	SEG	POS
max pooling	95.54	91.20
min pooling	95.40	91.02
average pooling	95.83	**91.35**
self-attention pooling	95.76	91.28
LSTM-Minus	95.67	91.22
combination	**95.86**	91.30

Table III
THE INFLUENCES OF DIFFERENT COMPOSITION APPROACHES FOR $\text{WORD}_{\text{CHARACTER}}$.

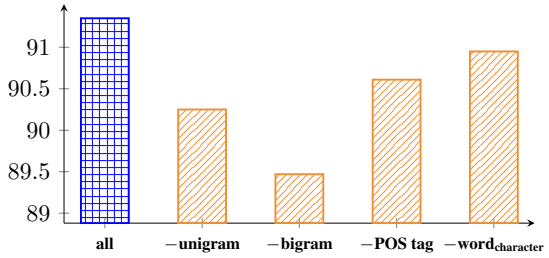


Figure 2. Feature ablation results on the CTB6 development corpus, where the Y-axis shows the F-measures of POS.

As the major computation cost lies in the calculation of encoder and decoder LSTMs, while the other parts affect much less, for example the output layer of next action prediction. Thus the speed of the joint model is comparable with the baseline word segmentation model, saving the cost of the pipeline POS tagging system. We perform all experiments on Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz with a single thread. The proposed model reaches speeds of 40.3 and 84.7 sentences per second for training and testing on average on the CTB datasets, respectively. For the pipeline system, the average training and testing speeds are 21.8 and 48.5 sentences per second, respectively.

D. Development Results

In order to better understand the proposed neural model for joint word segmentation and POS tagging, we conduct several development experiments on the CTB6 dataset.

Word Representation ($\text{word}_{\text{character}}$). First, we concern the word representation methods based on its covering characters. We have introduced five commonly-used approaches to obtain a word representation, including max, min, average and self-attention pooling and LSTM-Minus, as described in the `Decoder` subsection. We compare the five methods and further combine them to investigate the best way of word representation. The combination is achieved by simple vector concatenation of the five resulting representations.

Table III shows the results based on F-measures, where no external character embeddings are used in the experiments. As shown, except the min pooling, all the other methods are comparable without significant differences in performances. Average pooling gives a slightly larger number on the final POS tagging F-measure, and the combination of the five approaches does not bring improved results. Thus we adopt average pooling as the final composition method.

Feature Ablation. Second, we examine the effectiveness of various features, including character unigrams, character bigrams, POS tags and the word-level features composed from character-level BI-LSTM ($\text{word}_{\text{character}}$) as mentioned above. We conduct experiments with no external embeddings. Figure 2 shows the feature ablation results. According to the results, the character bigrams are most effective, without which the joint POS tagging performance decreases by close to 2.0% , and all the other features are also important and can bring improved performances.

Model	P	R	F
SEG			
no external embeddings	95.76	95.90	95.83
+basic embeddings	96.30	96.34	96.32
+word-context embeddings	**97.03**	**96.84**	**96.93**
POS			
no external embeddings	91.28	91.42	91.35
+basic embeddings	92.48	92.44	92.46
+word-context embeddings	**93.23**	**93.04**	**93.14**

Table IV
THE INFLUENCES OF DIFFERENT PRETRAINED EMBEDDINGS.

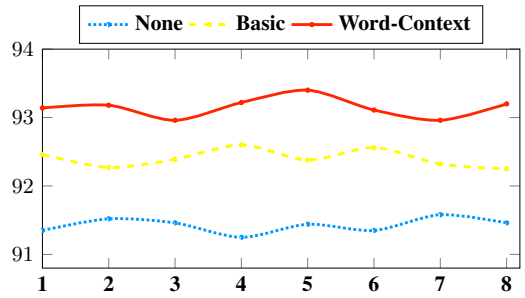


Figure 3. The influence of beam-search on the CTB6 development corpus, where the Y-axis shows the F-measures of POS.

Influences of Embeddings. Thirdly, we observe the influences of using different character embeddings. As illustrated in the `Encoder` subsection by the `Pretrained Embeddings` part, we have suggested two kinds of pretrained embeddings for character unigrams and bigrams. Here we verify the effectiveness of the pretrained embeddings. Table IV shows the results, which demonstrates that the pretrained embeddings are highly effective. When the **basic embeddings** are exploited, the F-measure values for both segmentation and POS tagging increase significantly. When the **word-context embeddings** are used, the model performances are boosted further. The results indicate that the external pretrained embeddings are highly important in our model, and the **word-context embeddings** are much better.

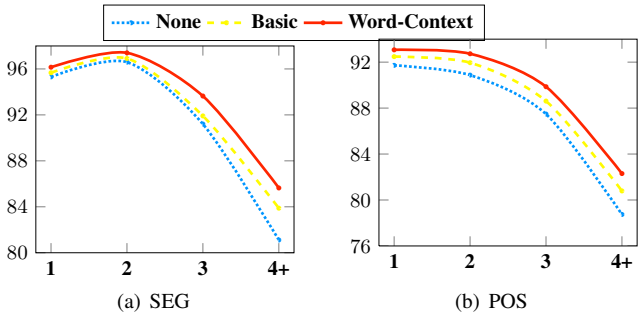
Beam Search. Beam search has been a widely-adopted technique in Seq2Seq models to improve system performances [34]. We verify its effectiveness in our proposed models. As shown in Figure 3, we find beam search does not bring any significant improvements. The main reason could be due to that the output transition action number is relatively small. Actually, the same observation is found in Dyer et al. (2015) [36], who propose a transition-based dependency parsing model that can be also formalized as Seq2Seq learning.

Model	CTB5		CTB6		CTB7		PKU		NCC	
	SEG	POS	SEG	POS	SEG	POS	SEG	POS	SEG	POS
Our Model (No External Embeddings)	97.69	94.16	95.37	90.83	95.32	90.25	95.22	92.62	93.97	89.47
Pipeline (No External Embeddings)	97.15	93.72	94.85	90.08	94.71	89.56	94.86	91.84	93.54	88.52
Our Model (Basic Embeddings)	97.93	94.44	95.78	91.79	95.77	91.12	95.82	93.42	94.52	89.82
Pipeline (Basic Embeddings)	97.50	94.01	95.58	91.35	95.36	90.78	95.55	93.00	94.17	89.25
Our Model (Word-context Embeddings)	**98.50**	**94.95**	**96.36**	**92.51**	**96.25**	**91.87**	**96.35**	**94.14**	**95.30**	**90.42**
Pipeline (Word-context Embeddings)	98.34	94.52	96.21	91.99	96.01	91.36	96.17	93.87	94.88	89.92
ZPar (discrete) [13]	97.68	93.73	95.40	90.84	95.12	90.06	95.09	92.40	93.30	87.44
Wang et al. (2011) [51]	98.11	94.18	95.79	91.12	95.65	90.46	–	–	–	–
Zhang et al. (2014) [32]	97.84	94.62	95.56	91.39	95.51	90.76	–	–	–	–
Chen et al. (2017) [25]	–	93.19	–	–	–	–	–	–	–	88.76
Kurita et al. (2017)[26]	98.37	94.83	–	–	96.23	91.25	–	–	–	–
Shao et al. (2017) [27]	97.89	94.07	–	–	–	–	–	–	–	–

Table V
FINAL RESULTS ON THE TEST DATASETS OF CTB5, CTB6, CTB7, PKU AND NCC.

Embeddings	SEG		TAG	
	R_{iv}	R_{oov}	R_{iv}	R_{oov}
No external embeddings	96.64	73.79	92.52	62.34
Basic embeddings	96.69	78.65	93.00	69.57
Word-context embeddings	**96.98**	**82.52**	**93.45**	**73.57**

Table VI
THE PERFORMANCES OF IV AND OOV WORDS.



(a) SEG (b) POS

Figure 4. F-measures against word length.

E. Final Results

Table V shows the final results on the test datasets of CTB5, CTB6, CTB7, PKU and NCC. We present the performances of the Seq2Seq neural model by using no external embeddings, the basic character embeddings and the word-context embeddings, respectively. According to the results, we can see that pretrained embeddings are very useful, and the word-context embeddings achieve the best performances on all three datasets, which is consistent with the findings of the development experiments. We compare the proposed neural joint models with the corresponding discrete baseline system **ZPar** [13]. Under a fair purely-supervised setting, the neural joint model without any pretrained embeddings achieves comparable or better performances on the five datasets compared with this baseline. When external embeddings are used, the neural joint model gives significantly better performances (p-value is below 10^{-5} by using pairwise t-test).

In addition, we compare the proposed joint model with the corresponding pipeline system as well. As shown in Table V, the neural joint model outperforms the pipeline system under all conditions, demonstrating the effectiveness of the joint framework. The difference is highly significant when we do not use any kind of pretrained embeddings, while the gap becomes smaller when basic and word-context embeddings are exploited as augmented information. The observation indicates that the pretrained embeddings have included syntactic POS information already, which is consistent with the findings in Mikolov et al. (2013) [18]. In the joint model, segmentation and POS-tagging are capable of full interaction, not only reducing the error propagation, but also improving segmentation performances in most cases by utilizing syntactic POS tag information, thus bring better performances for both word segmentation and POS tagging.

We also compare the proposed Seq2Seq neural model with other work in the literature. As shown in Table V, our model with the basic embeddings is better than Wang et al. (2011) [51], which is a semi-supervised model by using traditional handcrafted features, better than Zhang et al. (2014) [32], which is a joint model for Chinese word segmentation, POS tagging and dependency parsing, and also better than Shao et al. (2017) [27], which is a CRF-based neural sequence labeling model with additional radical and orthographical features of Chinese characters besides the character-level embeddings. Our model with the word-context embeddings outperforms the word-context model of Kurita et al. (2017) [26], which exploits large-scale segmented corpus to pretrain their embddings.

F. Discussion

In this subsection, we conduct analysis work to examine the proposed Seq2Seq neural model. All the work is performed on the CTB6 test dataset.

IV & OOV. First, we examine the influences of in-vocabulary (IV) and out-of-vocabulary (OOV) words in our proposed neural model. We examine the model with no external embeddings, basic embeddings and word-context embeddings, respectively, computing their recalls with respect to IV and OOV words. Table VI shows the results. We can see that the recalls of IV words are consistently much higher than the OOV recalls, which is reasonable since unseen words are more difficult to be recognized. In addition, by using pretrained character embeddings, both the IV and OOV recalls increase, and the improvements of the OOV recalls are more

Pipeline v.s. Joint	
Pipeline	**深爱*(deeply love)*|*VV*** 汉代(Han dynasty)|NR 文采(literary)|NN 的(de)|DEC 柯庆明(Ke Qingming)|NR
Joint	**深*(deeply)*|*AD* 爱*(love)*|*VV*** 汉代(Han dynasty)|NR 文采(literary)|NN 的(de)|DEC 柯庆明(Ke Qingming)|NR
Pipeline	中国(China)|NR 对(to)|P 外(foreign)|NN 开放(open)|VV 始于(start by)|VV 八十*(eighty)*|*CD* 年代*(age)*|*NN*
Joint	中国(China)|NR 对(to)|P 外(foreign)|NN 开放(open)|VV 始于(start by)|VV 八十年代*(eighties)*|*NT*
Joint: None embeddings v.s. Basic embeddings v.s. Word-context embeddings	
None	**新*(new)*|*AD* 科*(branch)*|*NN*** 诺贝尔(Nobel)|NR 文学(literature)|NN 奖(prize)|NN 得主(winner)|NN
Basic	**新科*(new)*|*NN*** 诺贝尔(Nobel)|NR 文学(literature)|NN 奖(prize)|NN 得主(winner)|NN
Word-context	**新科*(new)*|*JJ*** 诺贝尔(Nobel)|NR 文学(literature)|NN 奖(prize)|NN 得主(winner)|NN
None	深圳(Shenzhen)|NR 保税区**绘*(-unknown-)*|*NN* 就*(towards)*|*VV*** 新(new)|JJ 规划(plan)|NN
Basic	深圳(Shenzhen)|NR 保税区(Free Trade Zone)|NN **绘 *(draw)*|*VV* 就*(towards)*|*P*** 新(new)|JJ 规划(plan)|NN
Word-context	深圳(Shenzhen)|NR 保税区(Free Trade Zone)|NN **绘就*(draw already)*|*VV*** 新(new)|JJ 规划(plan)|NN

Table VII
CASE STUDIES OF THE PROPOSED SEQ2SEQ NEURAL MODEL.

significant. The word-context embeddings can bring increases close to 10% on the segmentation and POS recalls.

Word Length. Second, we investigate the proposed neural model by the capability of modeling words of different lengths, following the work of Zhang et al. (2016) [21]. Figure 4 shows the results. For word segmentation, under all settings, the performances arrive at the best when the length is 2, and drop slightly when the length is 1, and become significantly worse when the length is larger than 2. The phenomenon is consistent with the finding of Zhang et al. (2016) [21]. While for POS tagging, the trendy is a little different, the performances decrease consistently when the length increases. According to the observation, the single-character words are more difficult to be recognized, while their POS tags are easier to be identified, which demonstrates the less ambiguities of single-character words. This is reasonable as the potential POS information of a composed word is relatively richer than its components individually.

Case Studies. We show several case studies to demonstrate the advantages of our neural joint model. We conduct the analysis by two aspects. On the one hand, we show how the proposed model benefits from the joint modeling framework. The upper of Table VII offers two examples. The examples are selected from our joint model with word-context embeddings and the pipeline model described in the previous subsection including the same embeddings. According to the CTB segmentation guideline[4], sometimes we need to refer the POS tag of a subword in order to decide whether it can be regarded as a full word. For example, in the first example, both "深(deeply)" and "爱(love)" should be treated as full words, based on the rule of "AD+V"(adverb + verb). While in the second example, "八十(eighty)" and "年代(age)" should be combined together as one full word, based on the rule of "CD+N" (number + noun). For these cases, the joint model is better as it can integrate POS information for word segmentation.

On the other hand, we show several examples to express the differences among our neural joint model with different character embeddings. As shown by the below part of Table VII, in the first case, the neural joint model without external embeddings is incapable of handling OOV word "新科(new)", and it segments the word into two separate words, since "新(new)" and "科(branch)" are both common words. Although the model with basic embeddings correctly recognizes it as a single word, it regards the word as noun. As an unknown word without background knowledge, this is reasonable because the suffix "科(branch)" can be treated as noun. Only the model with word-context embeddings correctly gets the word, tagging it as an adjective word, which is possibly due to that word-context embeddings have already learned the required background knowledge. The second example also verifies that pretrained external embeddings are better, and word-context embeddings can capture more background information, for example, the meaning of "accomplish" or "already" for the Chinese character "就(jiu)".

[4]http://www.cs.brandeis.edu/~clp/ctb/segguide.3rd.ch.pdf

IV. RELATED WORK

State-of-the-art Chinese word segmentation models can be categorized into two types: character-based [1] and word-based models [4]. Both two kinds of models have been investigated intensively under traditional statistical and neural network settings. The character-based models are usually solved by conditional random filed (CRF) [3], [24], [28], [37], using features from character unigrams and bigrams. Comparing with character-based models, the word-based models are capable of using word-level features, which are another source of useful features [20], [21], [39], [52].

Our work is closely related to the work of Zhang et al. (2016) [21], which is a transition-based neural model for word segmentation. Our transition system can be regarded as an extension of the work, by integrating POS tagging at the same time. Our work is also significantly different in that we exploit a Seq2Seq framework to obtain strong performances, using only a greedy search strategy based on well-designed neural network structures.

POS tagging is commonly treated as a sequence labeling problem, and is generally solved by CRF or transition-based models [6]–[8], [53]. Features can be slightly different among different languages. For Chinese, word-level and character-level features are both important [5], [31]. Recently, neural network models are able to achieve better performances than traditional statistical models based on discrete features, by using bi-directional LSTM to extract features [22], [23].

For Chinese, joint word segmentation and POS tagging has shown improved performances compared with the pipeline systems [9]–[13], [54], as the two tasks are closely related, and on the other hand the joint models can reduce error propagation [55], [56]. Our work is mainly inspired by the great success of these studies. We propose a Seq2Seq neural model for the joint task, since neural network models have shown better performances on word segmentation.

There have been several neural based joint models for word segmentation and POS tagging [24]–[26]. Almost all studies exploit character-level CRF framework to achieve the goal, which is unable to incorporate word-level information. Only one exception is the work of Kurita et al. (2017) [26], which performs the best among previous studies on the CTB5 dataset. The work exploits a transition-based method as well but requires much effort on feature engineering (more than 40 atomic features are used in the model). Our work aims to present a simpler yet more effective model for the joint task. We exploit a Seq2Seq framework and give the top performances among all datasets used in our experiments.

The goal of Seq2Seq learning is to predict a sequence of target symbols conditioned on an observed sequence [15]. The framework has been successfully applied on a number of NLP tasks due to its simplicity and efficiency, including machine translation [33], dialogue [16], syntax parsing [30], question answering [17] and summarization [57]. In this work, we apply the Seq2Seq framework on joint word segmentation and POS tagging, by using a transition system to convert the problem into predicting a sequence of symbols (transition actions).

V. Conclusions

We proposed a simple and effective Seq2Seq neural model for joint Chinese word segmentation and POS tagging. The model is built based on a well defined transition system for the joint task, which transforms the decoding process into predicting a sequence of transition actions. The key features of the neural model are based on two LSTMs, one of which is the bi-directional LSTM over the input character sequence, and the other is a left-to-right LSTM over the historical word-tag pairs that can be directly obtained by the transition system. To better represent characters, we presented the word-context character embeddings. We conducted experiments on five standard datasets to evaluate our joint models. Experimental results demonstrated that the proposed model is highly competitive. By using the word-context character embeddings, the Seq2Seq neural model is able to achieve the best-reported performances on all the five datasets.

References

[1] N. Xue, "Chinese word segmentation as character tagging," *International Journal of Computational Linguistics and Chinese Language Processing*, vol. 8(1), 2003.

[2] H. T. Ng and J. K. Low, "Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based?" in *Proceedings of EMNLP 2004*, 2004, pp. 277–284.

[3] F. Peng, F. Feng, and A. McCallum, "Chinese segmentation and new word detection using conditional random fields," in *Proceedings of Coling 2004*, Aug 23–Aug 27 2004, pp. 562–568.

[4] Y. Zhang and S. Clark, "Chinese segmentation with a word-based perceptron algorithm," in *Proceedings of the 45th ACL*, June 2007, pp. 840–847.

[5] W. Sun and H. Uszkoreit, "Capturing paradigmatic and syntagmatic lexical relations: Towards accurate chinese part-of-speech tagging," in *ACL*, July 2012, pp. 242–252.

[6] Z. Li, J. Chao, M. Zhang, W. Chen, M. Zhang, G. Fu, Z. Li, J. Chao, M. Zhang, W. Chen *et al.*, "Coupled pos tagging on heterogeneous annotations," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 3, pp. 557–571, 2017.

[7] M. Collins, "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms," in *EMNLP*, July 2002, pp. 1–8.

[8] L. Yang, M. Zhang, Y. Liu, M. Sun, N. Yu, and G. Fu, "Joint pos tagging and dependency parsing with transition-based neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1–1, 2017.

[9] W. Jiang, L. Huang, Q. Liu, and Y. Lü, "A cascaded linear model for joint chinese word segmentation and part-of-speech tagging," in *Proceedings of ACL*, June 2008, pp. 897–904.

[10] Y. Zhang and S. Clark, "Joint word segmentation and POS tagging using a single perceptron," in *Proceedings of ACL-08: HLT*, June 2008, pp. 888–896.

[11] C. Kruengkrai, K. Uchimoto, J. Kazama, Y. Wang, K. Torisawa, and H. Isahara, "An error-driven word-character hybrid model for joint chinese word segmentation and pos tagging," in *ACL*, August 2009, pp. 513–521.

[12] T. Qian, Y. Zhang, M. Zhang, Y. Ren, and D. Ji, "A transition-based model for joint segmentation, pos-tagging and normalization," in *Proceedings of the 2015 EMNLP*, 2015, pp. 1837–1846.

[13] Y. Zhang and S. Clark, "A fast decoder for joint word segmentation and POS-tagging using a single discriminative model," in *Proceedings of the EMNLP*, October 2010, pp. 843–852.

[14] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *JMLR*, vol. 12, pp. 2493–2537, 2011.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[16] O. Vinyals and Q. Le, "A neural conversational model," *arXiv preprint arXiv:1506.05869*, 2015.

[17] J. Yin, X. Jiang, Z. Lu, L. Shang, H. Li, and X. Li, "Neural generative question answering," in *IJCAI*, 2016.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[19] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[20] D. Cai and H. Zhao, "Neural word segmentation learning for chinese," in *Proceedings of ACL 2016*, 2016.

[21] M. Zhang, Y. Zhang, and G. Fu, "Transition-based neural word segmentation," in *Proceedings of ACL*, Berlin, Germany, 2016, pp. 421–431.

[22] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[23] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *Proceedings of the 54th ACL*, Berlin, Germany, August 2016, pp. 1064–1074. [Online]. Available: http://www.aclweb.org/anthology/P16-1101

[24] X. Zheng, H. Chen, and T. Xu, "Deep learning for Chinese word segmentation and POS tagging," in *Proceedings of the 2013 Conference on EMNLP*, Seattle, Washington, USA, October 2013, pp. 647–657. [Online]. Available: http://www.aclweb.org/anthology/D13-1061

[25] X. H. Xinchi Chen, Xipeng Qiu, "A feature-enriched neural model for joint chinese word segmentation and part-of-speech tagging," in *Proceedings of the IJCAI*, 2017.

[26] S. Kurita, D. Kawahara, and S. Kurohashi, "Neural joint model for transition-based chinese syntactic analysis," in *Proceedings of the 55th ACL*, July 2017, pp. 1204–1214.

[27] Y. Shao, C. Hardmeier, J. Tiedemann, and J. Nivre, "Character-based joint segmentation and pos tagging for chinese using bidirectional rnn-crf," *arXiv preprint arXiv:1704.01314*, 2017.

[28] X. Chen, X. Qiu, C. Zhu, P. Liu, and X. Huang, "Long short-term memory neural networks for chinese word segmentation," in *Proceedings of EMNLP*, September 2015, pp. 1197–1206. [Online]. Available: http://aclweb.org/anthology/D15-1141

[29] H. Zhou, Z. Yu, Y. Zhang, S. Huang, X. Dai, and J. Chen, "Word-context character embeddings for chinese word segmentation," *Proceedings of EMNLP*, 2017.

[30] L. Kong, C. Alberti, D. Andor, I. Bogatyy, and D. Weiss, "Dragnn: A transition-based framework for dynamically connected neural networks," *arXiv preprint arXiv:1703.04474*, 2017.

[31] Z. Li, M. Zhang, W. Che, T. Liu, W. Chen, and H. Li, "Joint models for chinese pos tagging and dependency parsing," in *EMNLP*, July 2011, pp. 1180–1191.

[32] M. Zhang, Y. Zhang, W. Che, and T. Liu, "Character-level chinese dependency parsing," in *Proceedings of the 52nd ACL*, June 2014, pp. 1326–1336.

[33] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[34] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[35] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[36] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," in *ACL*, July 2015, pp. 334–343.

[37] W. Pei, T. Ge, and B. Chang, "Max-margin tensor neural network for chinese word segmentation," in *Proceedings of the 52nd ACL*, Baltimore, Maryland, June 2014, pp. 293–303. [Online]. Available: http://www.aclweb.org/anthology/P14-1028

[38] J. Yang, Y. Zhang, and F. Dong, "Neural word segmentation with rich pretraining," in *ACL*, July 2017, pp. 839–849.

[39] Y. Liu, W. Che, J. Guo, B. Qin, and T. Liu, "Exploring segment representations for neural segmentation models," in *Proceedings of IJCAI 2016*, 2016.

[40] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of the 52nd ACL*, June 2014, pp. 302–308.

[41] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.

[42] W. Wang and B. Chang, "Graph-based dependency parsing with bidirectional lstm," in *ACL*, August 2016, pp. 2306–2315.

[43] M. Zhang, G. Fu, and N. Yu, "Segmenting chinese microtext: Joint informal-word detection and segmentation with neural networks," in *Proceedings of IJCAI 201t*, 2017.

[44] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *EMNLP*, October 2014, pp. 740–750.

[45] M. Zhang and Y. Zhang, "Combining discrete and continuous features for deterministic transition-based dependency parsing," in *Proceedings of the 2015 EMNLP*, September 2015, pp. 1316–1321. [Online]. Available: http://aclweb.org/anthology/D15-1153

[46] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins, "Globally normalized transition-based neural networks," in *ACL*, August 2016, pp. 2442–2452.

[47] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[48] G. Jin and X. Chen, "The fourth international chinese language processing bakeoff: Chinese word segmentation, named entity recognition and chinese pos tagging," in *Proceedings of the sixth SIGHAN workshop on Chinese language processing*, 2008.

[49] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "Part-of-speech tagging with bidirectional long short-term memory recurrent neural network," *arXiv preprint arXiv:1510.06168*, 2015.

[50] C. D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *Proceedings of the 31st ICML*, 2014, pp. 1818–1826.

[51] Y. Wang, J. Kazama, Y. Tsuruoka, W. Chen, Y. Zhang, and K. Torisawa, "Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data," in *IJCNLP*, November 2011, pp. 309–317.

[52] X. Sun, Y. Zhang, T. Matsuzaki, Y. Tsuruoka, and J. Tsujii, "A discriminative latent variable chinese segmenter with hybrid word/character information," in *NAACL*, June 2009, pp. 56–64.

[53] K. Toutanova, D. Klein, C. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network." in *Proceedings of HLT-NAACL 2003*, 2003.

[54] X. Qian, Q. Zhang, Y. Zhou, X. Huang, and L. Wu, "Joint training and decoding using virtual nodes for cascaded segmentation and tagging tasks," in *Proceedings of the EMNLP*, Cambridge, MA, October 2010, pp. 187–195. [Online]. Available: http://www.aclweb.org/anthology/D10-1019

[55] Z. Li, M. Zhang, W. Che, T. Liu, and W. Chen, "Joint optimization for chinese pos tagging and dependency parsing," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 274–286, 2014.

[56] D. Tang, B. Qin, F. Wei, L. Dong, T. Liu, and M. Zhou, "A joint segmentation and classification framework for sentence level sentiment classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 11, pp. 1750–1761, 2015.

[57] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th ACL*, Vancouver, Canada, July 2017, pp. 1073–1083.