Exploring Segment Representations for Neural Semi-Markov Conditional Random Fields

Yijia Liu, Wanxiang Che, Bing Qin, and Ting Liu

***Abstract*—Many problems in natural language processing (NLP) can be cast as the problem of segmenting a sequence. In this article, we combine the semi-Markov conditional random fields (semi-CRF) with neural networks to solve NLP segmentation problems. We focus on the segment representation in neural semi-CRF which is important to the performance. Based on our preliminary work in Liu *et al.* [1], we represent a segment by both encoding the subsequence and embedding the segment string. We conduct a systematic study of the utility of various components in subsequence encoding and propose a method of constructing and deriving segment string embeddings. Extensive experiments on three typical segmentation problems, namely, shallow syntax parsing, named entity recognition, and Chinese word segmentation are conducted. The results show that we can achieve equally-performed subsequence encoding with a three times faster concatenation network compared to previous work. The results also show that the segment string embeddings help our neural semi-CRF model to achieve a macro-averaged error reduction of 13.15% over a strong baseline using deep contextualized embeddings and bidirectional long-short-term memory CRF, which also show the usefulness of semi-CRF even with contextualized embeddings. These results are competitive with the state-of-the-art segmentation systems.**

***Index Terms*—Neural network, semi-CRF, segment representation, shallow syntax parsing, named entity recognition, Chinese word segmentation.**

I. Introduction

MANY problems in natural language processing (NLP) involve segmenting and assigning tags to a sequence of *observations*, like shallow syntax parsing [2, chunking], named entity recognition [3, NER], opinion extraction [4], disfluency detection [5], and Chinese word segmentation [6], [7, CWS]. Properly representing the *segment* (i.e., subsequence), such as chunk in chunking, entity in NER, and Chinese word in CWS, is important for good *segmentation* performance. Sequence labeling models like conditional random fields [8, CRFs] are widely used for these problems, which label an individual observation with a segment boundary. Compared with sequence labeling that labels the boundary as a proxy to the segment, models that directly encode a segment are attractive because they can effectively utilize the segment-level features like "entity length" in NER. The semi-Markov conditional random fields [9, semi-CRF] is one of the models that directly encode a segment. In a semi-CRF, given a sequence of observations, its conditional probability of a semi-Markov chain is explicitly modeled, in which each state corresponds to the segment. The practice of modeling a state in the semi-Markov chain makes semi-CRF a natural choice for segmentation problems.

To achieve good segmentation performance, conventional semi-CRFs require carefully hand-crafted features [4], [6], [9]. However, manually designing the features is tedious and incomplete, which call for automatic feature extraction. With the success of representation learning in NLP, research efforts have been made to automatically extract features with neural network in semi-CRFs. Kong *et al.* [10] proposed a segmental recurrent neural network (SRNN) to embed the subsequence of input as features, and the input vectors are context-dependent via a bidirectional long-short-term-memory [11, biLSTM] encoding. They tested their model on handwriting recognition and CWS. Zhuo *et al.* [12] proposed a gated recursive convolutional neural network (grConv) to encode the subsequence of context-independent input vectors, and reported improved NER performance. Both these works emphasized the importance of representing a segment with neural network. However, a systematic study of the utility of various components which comprise the neural semi-CRFs (see Fig. 2) under the same experimental settings was missing. In addition, the whole segment, either used as an input feature or as a lexicon indicator, can be important to evaluate a segmentation. Kong *et al.* [10] and Zhuo *et al.* [12] both treated the segment as a subsequence of input, the idea of representing the segment with respect of its string form was less mentioned.

In this paper, we conduct a thorough study on the problem of representing a segment in neural semi-CRFs. Our efforts of segment representation fall into two folds. In the first fold, we represent the segment by *encoding the subsequence of input*. We study the necessity of context encoding for the input representation. We also propose four network architectures to encode the subsequence as alternatives to SRNN and grConv. In the second fold, we represent the segment by *embedding the segment string* and propose to derive these *segment string embeddings* from automatically segmented data. Extensive experiments on three typical NLP segmentation tasks–chunking, NER, and CWS are conducted. The results show that context encoding plays an important role in neural semi-CRFs. Our concatenation alternative achieves comparable performance with SRNN and grConv,

Manuscript received January 18, 2019; revised July 5, 2019, October 21, 2019, and December 22, 2019; accepted December 30, 2019. Date of publication January 8, 2020; date of current version February 11, 2020. This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 61976072, Grant 61632011, and Grant 61772153. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Eric Fosler-Lussier. *(Corresponding author: Wanxiang Che.)*

The authors are with the Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology, Harbin 150001, China (e-mail: yjliu@ir.hit.edu.cn; car@ir.hit.edu.cn; qinb@ir.hit.edu.cn; tliu@ir.hit.edu.cn).

Digital Object Identifier 10.1109/TASLP.2020.2964960

while running more than 3 times faster. The results also show that our neural semi-CRF benefits from segment string embeddings, with a macro-averaged error reduction of 13.15% over a strong baseline using deep contextualized embeddings [13] and biLSTM CRF [14]. In the evaluated tasks, our model achieves competitive performance with the state-of-the-art (SOTA) segmentation systems and is 0.43 points lower than the aggregation of the results of these systems. Major contributions of this paper include:

- We thoroughly study the subsequence encoding for segment representation and propose a simple and accurate concatenation network (§IV-A). We also propose to use segment string embeddings as the segment representation and find that it leads to consistent improvements (§IV-B).
- We conduct extensive experiments (§V) to reveal the important components in neural semi-CRFs. The experimental results show the importance of representing context and the effectiveness of our concatenation segment encoder. The results also show that consistent improvements are achieved with our segment string embeddings over a strong baseline using deep contextualized embeddings and biLSTM-CRF. The results are competitive with the SOTA systems.

Some preliminary results have been reported at our previous work [1]. This paper extends our previous work with a deep study on the effect of different components in neural semi-CRF (§IV-A) and emphasizes the importance of context encoding. This finding also inspired us to enhance the corresponding encoding with the deep contextualized word embeddings (§IV-A1). In addition to the CNN and concatenation network, we proposed two more networks (§IV-A3) and introduced grConv into our comparison (§V-B). An additional study on the usage of segment embedding alone (§IV-B) presents a thorough understanding of the segment embeddings. Recent study [15] on the non-deterministic nature of neural network training suggests conducting of multi-seeded experiment runs and reporting the average score. We follow this idea and re-run the experiments in our previous work. The new results challenge our former discussions on the use of different baseline segmentors by presenting negligible performance gaps. Therefore, we remove the corresponding part in this version. A new segmentation data—CoNLL00 chunking dataset was introduced to extend our evaluation (§V-A). Additional analysis shows the segment diversity as a factor for segment string embedding to work well (§V-D). Our new PyTorch implementation is available at https://github.com/Oneplus/semiCRF.

II. Problem Definition

Fig. 1 shows examples of NER and CWS. For the input word sequence in the NER example, its segments (*"Michael Jordan":PER, "is":NONE, "a":NONE, "professor":NONE, "at":NONE, "Berkeley":ORG*) reveal that "Michaels Jordan" is a person name and "Berkeley" is an organization. In the CWS example, the subsequences ("浦东/Pudong," "开发/development," "与/and," "建设/construction") of the input characters are recognized as words. Both NER and CWS take an input sequence and segment it into disjoint subsequences.

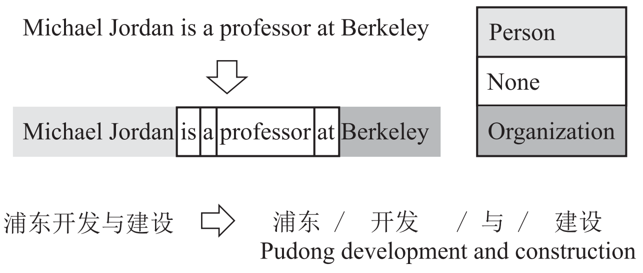


Fig. 1. Examples for named entity recognition (above) and Chinese word segmentation (below).

Formally, for an input sequence $\mathbf{x} = \langle x_1, \ldots, x_{|\mathbf{x}|}\rangle$ of length $|\mathbf{x}|$, a *segment* of $\mathbf{x}$ is defined as (u, v, y) which means the subsequence $\langle x_u, \ldots, x_v\rangle$ is associated with label y. A *segmentation* of $\mathbf{x}$ is a *segment* sequence $\mathbf{s} = \langle s_1, \ldots, s_{|\mathbf{s}|}\rangle$, where $s_j = (u_j, v_j, y_j)$ and $u_{j+1} = v_j + 1$. Given an input sequence $\mathbf{x}$, the segmentation problem can be defined as the problem of finding the most probable segment sequence $\mathbf{s}$.

We also define the *segment string* $x_{[u:v]}$ of a segment (u, v, y) as a string concatenation of the tokens in the segment with a special delimiter. Taking the NER example in Fig. 1 as an illustration, the subsequence of the segment $(1, 2, \text{PER})$ is ("Michael," "Jordan") while the corresponding segment string is "Michael_Jordan".

III. Neural Semi-CRFs

The semi-CRF (see Fig. 2) models the conditional probability of $\mathbf{s}$ on $\mathbf{x}$ as

$$p(\mathbf{s} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\{W \cdot G(\mathbf{x}, \mathbf{s})\},$$

where $G(\mathbf{x}, \mathbf{s})$ is the feature vector, W is the weight vector, and $Z(\mathbf{x}) = \sum_{\mathbf{s}' \in \mathbf{S}} \exp\{W \cdot G(\mathbf{x}, \mathbf{s}')\}$ is the normalization factor of all possible segmentations $\mathbf{S}$ over $\mathbf{x}$.

By restricting the scope of the features within a segment and ignoring the label transition between segments (i.e., zero-order semi-CRFs), $G(\mathbf{x}, \mathbf{s})$ can be decomposed as $G(\mathbf{x}, \mathbf{s}) = \sum_{j=1}^{|\mathbf{s}|} g(\mathbf{x}, s_j)$ where $g(\cdot, \cdot)$ maps a segment s_j into its representation. Such decomposition allows using an efficient dynamic programming algorithm for inference. To find the best segmentation in semi-CRFs, let α_j denote the log probability of the best segmentation ending at j, and α_j can be recursively calculated as

$$\boldsymbol{\alpha}_j = \max_{l=1\ldots L, y} \Psi(j - l, j, y) + \boldsymbol{\alpha}_{j-l-1}, \tag{1}$$

where L is the manually defined maximum segment length and $\Psi(j - l, j, y) = W \cdot g(\mathbf{x}, (j - l, j, y))$ is the transition weight for segment $(j - l, j, y)$.

Previous semi-CRF works [3], [4], [6], [9] parameterize $g(\mathbf{x}, s)$ as a sparse vector, each dimension of which represents the value of the feature function. Kong *et al.* [10] pioneered in parameterizing $g(\mathbf{x}, s)$ with an RNN (SRNN, see Fig. 2). They first obtain the vector sequence $\langle \mathbf{v}_1, \ldots, \mathbf{v}_{|\mathbf{x}|}\rangle$ of $\mathbf{x}$ with a lookup table $\phi(\cdot)$ that maps an input symbol into its vector. Then, they feed $\langle \mathbf{v}_1, \ldots, \mathbf{v}_{|\mathbf{x}|}\rangle$ into a biLSTM to obtain the

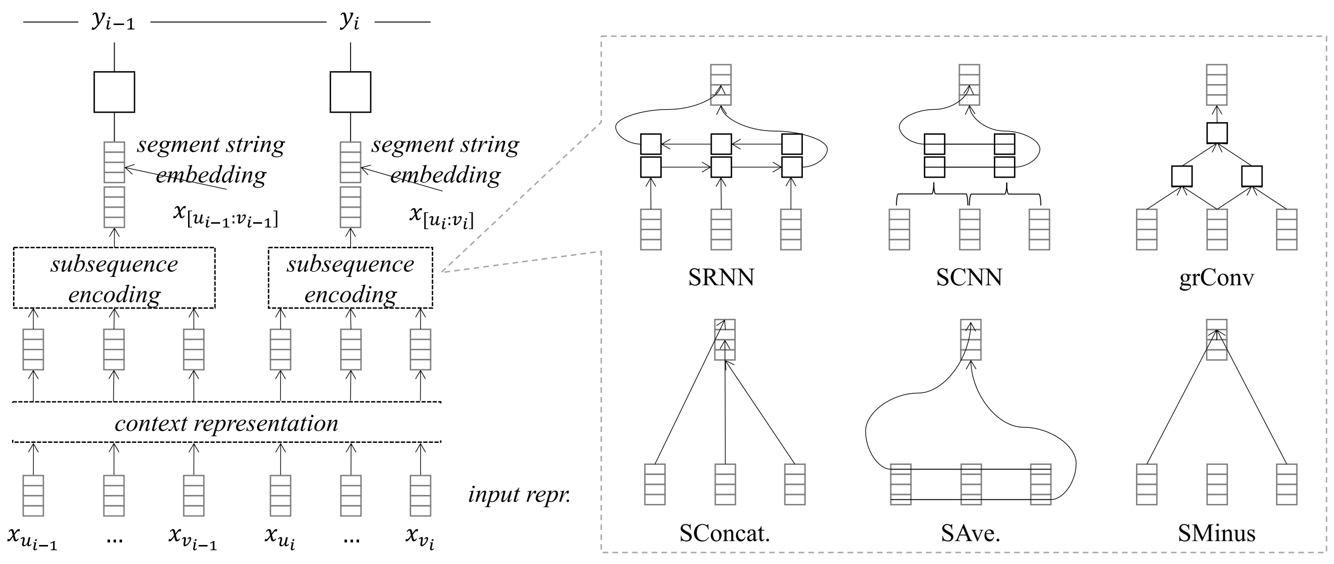


Fig. 2. An illustration for the framework of our neural semi-CRF model, along with the composition functions: SRNN, grConv, and our SAve (Eq. 5), SCNN (Eq. 6), SConcat (Eq. 7), and SMinus (Eq. 8).

contextual representation $\langle \mathbf{c}_1, \ldots, \mathbf{c}_{|\mathbf{x}|}\rangle$, Finally, they calculate the representation $g(\mathbf{x}, s_j)$ of a segment $s_j = (u_j, v_j, y_j)$ by feeding the subsequence $\langle \mathbf{c}_{u_j}, \ldots, \mathbf{c}_{v_j}\rangle$ int another biLSTM. The concatenation of the final hidden layers from forward and backward LSTMs is used as $g(\mathbf{x}, s)$. Zhuo *et al.* [12] proposed another network architecture — gated recursive convolutional neural network (grConv, see Fig. 2), which extracts the segment representation from a pyramid of representations recursively built from $\langle \mathbf{v}_{u_j}, \ldots, \mathbf{v}_{v_j}\rangle$.

Kong *et al.* [10] and Zhuo *et al.* [12] both showed the possibility of a reasonable segment representation with a neural network that encodes a subsequence of input vectors in semi-CRFs. However, their models diverged in the input vectors (Kong et. al's input vectors are context-dependent while Zhuo *et al.*'s are context-independent.) Moreover, only a limited set of networks was tried in these works. The whole segment was also not properly represented.

IV. Segment Representations for Neural Semi-CRFs

Representing a segment (i.e., parameterizing $g(\mathbf{x}, s)$) lies at the core of neural semi-CRFs. In this paper, we study the representation problem in two directions: 1) representing the segment by encoding a subsequence of input, and 2) representing the segment by embedding its string form.

For the segment representation by encoding the subsequence, three components comprise the whole model:

1) an *input representing component* (§IV-A1) that converts the input sequence $\mathbf{x} = \langle x_1, \ldots, x_{|\mathbf{x}|}\rangle$ into its context-independent vector sequence $\langle \mathbf{v}_1, \ldots, \mathbf{v}_{|\mathbf{x}|}\rangle$;
2) a *context representing component* (§IV-A2) that computes the context-dependent vector sequence $\langle \mathbf{c}_1, \ldots, \mathbf{c}_{|\mathbf{x}|}\rangle$ from $\langle \mathbf{v}_1, \ldots, \mathbf{v}_{|\mathbf{x}|}\rangle$;
3) a *subsequence representing component* (§IV-A3) that encodes the context-dependent subsequence $\langle \mathbf{c}_u, \ldots, \mathbf{c}_v\rangle$ of the segment (u, v, y) into a vector.

In this paper, we tried different ways of encoding inputs, context, and segments.

For the segment representation by embedding the segment string, a lookup table gives a vector representation to the string $x_{[u:v]}$ of a segment (u, v, y). The major problem is how to build segments and how to learn their embeddings. We explore different ways of building segments in Section IV-B.

We separate the segment representation into two types because the forms of their inputs are quite different (sequence vs string). The difference is analogous to the difference between character based word embedding method, such as FastText [16] and the word-type based word embedding method, such as Word2Vec [17]. Such difference has motivated the use of different word representation methods and hopefully will motivate the use of different segment representation methods.

A. Representing Segment by Encoding the Subsequence

1) Input Representation: Embeddings are the foundation of the neural network-based approach. In this paper, we implement it as a lookup table $\phi(\cdot)$ that stores the mapping between the string of the input observation and its vector representation (i.e., static embeddings).

In addition to the lookup table, deep contextualized embeddings as an emerging and effective technique have helped to improve a range of NLP models. In this paper, we also tried using one of the successful techniques – Embeddings from Language Models [13, ELMo] as the input representation. We use ELMo in a "feature" manner, where fixed vectors are extracted from the ELMo model trained on the unlabeled data.[1] Formally, the ELMo-based input representation is calculated as the layer-wise

[1]We use ELMo rather than Bidirectional Encoder Representations from Transformers [18, BERT] mainly because previous studies report the successful application of ELMo as feature while BERT was mainly used as initialization. Since we focus on the feature-based usage of the contextualized embeddings, we choose the ELMo in this paper and leave BERT for future study.

weighted summation of the concatenated ($\oplus$) hidden units $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$ from the forward LSTM and the backword LSTM, respectively. It can be described as

$$\phi^{(\text{ELMo})}(x_i) = \text{ELMo}(x_1, \ldots, x_{|\mathbf{x}|})_i = \gamma \sum_{j=1}^{3} s_j \cdot \left(\overrightarrow{\mathbf{h}}_{i,j} \oplus \overleftarrow{\mathbf{h}}_{i,j} \right), \tag{2}$$

where γ and s_j are trainable scalars and other parameters of $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$ are fixed. We encourage the reader of this paper to read the original ELMo paper [13] for more details.

We treat ELMo as an input representation, although it can generally be considered as utilizing context information. We argue that ELMo's representation of contextual information is learned over unlabeled data rather than a specific task and it is closer to static embeddings in this case.

2) Context Representation: Kong *et al.* [10] and Zhuo *et al.* [12] differ in the ways of representing context. Kong *et al.* [10] uses a biLSTM to obtain the contextual vector of an input observation, while Zhuo *et al.* [12] only uses the context-independent input vectors. The diversity of their approaches raises the question regarding whether or not encoding context matters in neural semi-CRF. In this paper, we minimize the efforts of feature engineering and follow Kong *et al.* [10], using the biLSTM-encoded context as the context representation. Formally, it can be described as

$$\langle \mathbf{c}_1, \ldots, \mathbf{c}_{|\mathbf{x}|} \rangle = \text{biLSTM}(\mathbf{v}_1, \ldots, \mathbf{v}_{|\mathbf{x}|}). \tag{3}$$

To test the necessity of context representation, we also tried only using the input vectors as the context representation, which means

$$\langle \mathbf{c}_1, \ldots, \mathbf{c}_{|\mathbf{x}|} \rangle = \langle \mathbf{v}_1, \ldots, \mathbf{v}_{|\mathbf{x}|} \rangle. \tag{4}$$

3) Segment Representation by Encoding Subsequence: After obtaining $\langle \mathbf{c}_1, \ldots, \mathbf{c}_{|\mathbf{x}|} \rangle$, we use a neural network to construct the representation subsequence $\langle \mathbf{c}_u, \ldots, \mathbf{c}_v \rangle$ of a segment (u, v, y) into a fixed-size vector of the segment. Due to the nature of the various lengths of a segment, such a neural network should have the ability to take subsequence of the various lengths. Kong *et al.* [10] handled the variant lengths with an RNN while Zhuo *et al.* [12] handled it with a gated recursive convolutional neural network. In the following section, we propose four alternative networks, and all of them are able to handle the variant length nature.

a) Segmental Average (SAve): The neural bag-of-words [19] is a sentence modeling method that sums up the vectors of words to form a fixed-size sentence vector. Cai and Zhao [20] also reported a positive result using a similar technique to model words in CWS. By utilizing its advantage over the variable length inputs, we propose to use average-pooling to form the segment representation $\mathbf{s}^{(\text{seq})}$ as

$$\mathbf{s}^{(\text{seq})} = \frac{1}{v - u + 1} \sum_{i=u}^{v} \mathbf{c}_i. \tag{5}$$

By using the average, we hope to eliminate the scale imbalance in $\mathbf{s}^{(\text{seq})}$ induced by the variant numbers of inputs.

b) Segmental Convolutional Neural Network (SCNN): The pooling mechanism in *SAve* fully ignores the order and relative position of subsequence of inputs within one segment. However, the positional information can be important to recognize a segment. Using a convolutional neural network [21, CNN] to encode the n-grams can be a remedy. In this paper, we use the temporal CNN that applies filter functions to "slide" over the input units in a segment and uses pooling to construct outputs of filter functions into the representation of a segment. The segment representation with a CNN can be formalized as

$$\mathbf{s}^{(\text{seq})} = \text{Conv}(\mathbf{c}_u, \ldots, \mathbf{c}_v). \tag{6}$$

In practice, naively applying filter functions to a segment is time-consuming for obtaining the representation. In this paper, we propose an approximation by applying filters to the whole input sequence and use the pooling function defined on a segment to collect segmental respresentation. This approximation speeds up the representation process and makes the representation of a segment aware of its surrounding context.

d) Segmental Concatenation (SConcat): To fully preserve the order of inputs in one segment, we propose to use the concatenation as the segment representation. To handle variable lengths of inputs, we make use of the maximum length L for the inference (Eq. 1) and pad with zero to transform the variable-length inputs into a fixed-size one as

$$\mathbf{s}^{(\text{seq})} = W^{(\text{sconcat})}(\mathbf{c}_u \oplus \cdots \oplus \mathbf{c}_v \oplus \underbrace{0 \oplus \cdots \oplus 0}_{L-(v-u+1)\text{ zeros}}), \tag{7}$$

where $W^{(\text{sconcat})} \in \mathcal{R}^{(L \times |\mathbf{c}|) \times |\mathbf{c}|}$ projects the concatenated vector into the low dimension. Concatenation preserves the positions of inputs and speeds up the segment representation process since it does not involve matrix multiplication.

Concatenation can be treated as a CNN with a filter of L-width. In this case, it lies on the other end of the spectrum compared with *SAve* with respect to preserving the input order.

c) Segmental Minus (SMinus): Previous syntax parsing works [22], [23] witness an interest in representing a segment by the subtraction between vectors of the head and tail words, especially when combined with biLSTM context encoding of the sentence. A similar effort has been made on Chinese word segmentation [24]. In this paper, we follow these works and represent the segment with the elementwise difference of context vectors of the segment boundaries, as

$$\mathbf{s}^{(\text{seq})} = (\mathbf{c}_u - \mathbf{c}_v). \tag{8}$$

As mentioned in Wang and Chang [22], the minus-based segment representation relies on the sequence modeling ability of LSTM. Therefore, we only apply *SMinus* to the biLSTM encoded context representation (Eq. 3) without considering that of Eq. 4.

B. Representing Segment by Embedding the Segment String

For a segmentation problem, a segment is generally considered more informative and less ambiguous than an individual word. Incorporating features that measure the whole segment, like [“Michael_Jordon” is in the gazetteer], has usually resulted in a performance improvement in previous semi-CRF works [4], [9]. Segment representations in Section IV-A only model the segment as a subsequence of input. It is expected that the segment embedding that encodes the entire string as a vector can be an effective approach for segment representation.

In this paper, we treat the segment string as a segment-level input and use a lookup table to map its string to the vector representation. Formally, we define the segment string embedding of $s = (u, v, y)$ as

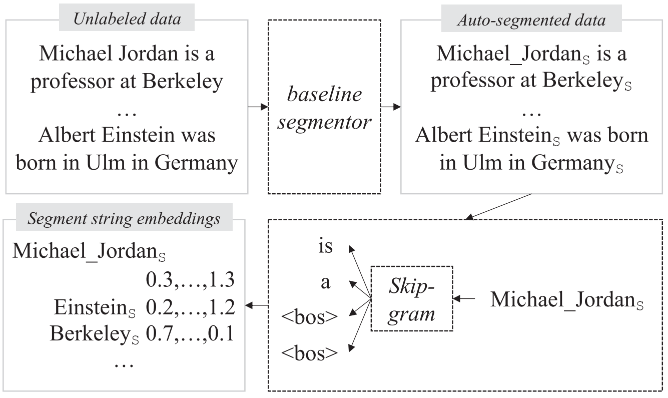
$$\mathbf{s}^{(\mathrm{emb})} = \phi^{(\mathrm{seg})}(x_{[u:v]}),$$

where $\phi^{(\mathrm{seg})}(\cdot)$ is the segment lookup table. It stores the mapping between the segment string, such as “Michael_Jordan,” and its vector. For the segment string not included in $\phi^{(\mathrm{seg})}(\cdot)$, we map it to a special unknown token. We use the superscript (seg) to differ the segment lookup table from the input lookup table $\phi(\cdot)$ in Section IV-A.

Using such segment representation in semi-CRF is straightforward. In this paper, we tried two different settings, including 1) using the segment string embedding as the only representation and 2) combining it with the representation calculated by composing input observations. The effect of the segment lookup table $\phi^{(\mathrm{seg})}(\cdot)$ resembles a group of lexicon-based segmentation methods, where the quality of lexicon greatly influences the model’s performance. Since one of the goals in a segmentation problem is to correctly recognize the boundaries, a string being included in the segment embeddings lookup table presents a strong clue for it to become a segment. Moreover, the practice that only collects the gold segments in the training data increases the risk of overfitting.

An ideal lookup table should recall more segments and also includes some ungrammatical segments that help to prevent overfitting the training data. We propose two methods for lookup table construction. One of them *constructs the lookup table from training data*, which uses the gold segment along with the most frequent ungrammatical segments in the training data.[2] Another method *constructs the lookup table from unlabeled data*, which uses a baseline segmentation model to analyze large-scale unlabeled data and constructs the lookup table with the automatically generated segments. The second method can be traced back to a line of semi-supervised research, which use features derived from auto-analyzed data to enhance the baseline model [25], [26]. The underlying intuition is that it encourages the model to consider the gold segments and the most probable ungrammatical segments according to the baseline.

In addition to the usage of lookup table construction, unlabeled data can also be used to derive segment string embeddings in an unsupervised fashion. In our second method, after obtaining the recognized segments on the unlabeled data, we concatenate the tokens in a segment to form a new entry and learn the segment string embeddings of these entries with the Word2Vec algorithm [17] on the same data. Fig. 3 illustrates our pipeline of mining the segments and learning the segment string embeddings from the unlabeled data. We first apply a baseline segmentor to the unlabeled data and obtain the automatically segmented data. The tokens of a recognized segment are concatenated and labeled with a special S tag.[3] The surrounding tokens are also fed into the Word2Vec algorithm to learn the segment string embeddings, but not collected into the lookup table as shown in Fig. 3. The unsupervisedly learned segment string embeddings can either be used as a fixed representation or as the initialization. When used as a fixed representation, the segments derived from the unlabeled data are transduced to the specific task, thus can mitigate the data sparsity of unseen segments in training.



Unlabeled data
Michael Jordan is a professor at Berkeley
…
Albert Einstein was born in Ulm in Germany
baseline segmentor
Auto-segmented data
Michael_Jordan$_S$ is a professor at Berkeley$_S$
…
Albert Einstein$_S$ was born in Ulm in Germany$_S$
Segment string embeddings
Michael_Jordan$_S$ 0.3,…,1.3
Einstein$_S$ 0.2,…,1.2
Berkeley$_S$ 0.7,…,0.1
…
is
a
<bos>
<bos>
Skip-gram
Michael_Jordan$_S$

Fig. 3. An illustration of our pipeline of mining segment and learning the segment string embeddings.

We need to note that using the baseline segmentor to mine segments does not resolve the sparsity problems. A typical case is that the larger the segments get, the harder it is to correctly mine their boundaries, and even harder to estimate the embeddings because of infrequent occurrence.[4] We conclude the sparsity problems as the diversity of segments of a certain dataset and study the relation between segment diversity and relative improvements in Section V-D.

C. Model Details

In this section, we describe the detailed architecture for our neural semi-CRFs.

1) Input Representation: To obtain the input representation, we use the technique in Dyer *et al.* [27] and separately use two parts of the input embeddings: the fixed pre-trained embeddings $\phi^{(\mathrm{fix})}(\cdot)$ and fine-tuned embeddings $\phi^{(\mathrm{tune})}(\cdot)$. Two of them are concatenated to form the final input unit representation as

$$\mathbf{v} = \phi^{(\mathrm{fix})}(x) \oplus \phi^{(\mathrm{tune})}(x). \qquad (9)$$

For chunking and NER, whose labeled data also has part-of-speech (POS) tags as inputs, we use the POS embeddings $\phi^{(\mathrm{tag})}(\mathrm{POS}(x))$ as an additional input representation and concatenate it to form $\mathbf{v}$.[5]

[2]By “ungrammatical segment,” we mean the segment of incorrect boundary.

[3]In this paper, we focus on the segment string embedding’s effect on identifying the boundary without considering the label. In practice, we remove the labels of the auto-recognized segments on the unlabeled data.

[4]The embedding of $\langle\mathrm{unk}\rangle$ are assigned to the un-recalled segments.

[5]The $\phi^{(\mathrm{tag})}(\mathrm{POS}(x))$ embedding is computed as a lookup table based on the POS tags and this table is randomly initialized during training.

TABLE I
THE HYPERPARAMETER SETTINGS

dim of fixed input unit embedding $\phi^{(\text{fix})}$ (for English)	300
dim of fixed input unit embedding $\phi^{(\text{fix})}$ (for Chinese)	100
dim of fine-tuned input unit embedding $\phi^{(\text{tune})}$	32
dim of ELMo $\phi^{(\text{ELMo})}$	1,024
dim of POS embedding $\phi^{(\text{tag})}$	12
dim of label embedding $\phi^{(\text{label})}$	20
dim of duration embeddings $\phi^{(\text{len})}$	4
biLSTM hidden layer size	128
number of layers in biLSTM (Eq. 3)	1
number of layers in biLSTM for SRNN	1
maximum segment length	10
segment string embedding $\phi^{(\text{seg})}$	50
batch size	32
dropout rate	0.1

After obtaining $\mathbf{v}$, we encode each the context of each word either with the function in Eq. 3 or the function in Eq. 4.

2) Segment Representation: Given a segment $s = (u, v, y)$, we feed its context representation $\langle \mathbf{c}_u, \ldots, \mathbf{c}_v \rangle$ into a segment encoding network $\mathcal{S}^{(\text{seq})}(\cdot)$ (i.e., Eqs. 5–8) to obtain $\mathbf{s}^{(\text{seq})}$ and feed the substring $x_{[u:v]}$ into $\phi^{\text{seg}}(\cdot)$ to obtain $\mathbf{s}^{(\text{emb})}$. $\mathbf{s}^{(\text{seq})}$ and $\mathbf{s}^{(\text{emb})}$ are either used separately or combined via concatenation. Two additional embedding functions $\phi^{(\text{len})}(\cdot)$ and $\phi^{(\text{label})}(\cdot)$ are used, where $\phi^{(\text{len})}(\cdot)$ is used to convert the discrete segment length to its length vector and $\phi^{(\text{label})}(\cdot)$ is used to convert label y to its vector. Finally, these vectors are concatenated and fed into a linear projection with ReLU activation to form the segment representation, which means

$$\mathbf{s}^{(\text{rep})} = \mathcal{S}^{(\text{seq})}(\mathbf{c}_u, \ldots, \mathbf{c}_v) \oplus \phi^{(\text{seg})}(x_{[u:v]}) \oplus \phi^{(\text{len})}(u - v) \oplus \phi^{(\text{label})}(y)$$

$$g(\mathbf{x}, s) = \text{ReLU}(W\mathbf{s}^{(\text{rep})} + b).$$

Throughout this paper, we use the same hyperparameters for different experiments as listed in Table I. We use 6 types of filters with widths from 1 to 6 for the CNN (Eq. 6).[6]

Training the parameters is accomplished by maximizing the log-likelihood

$$\arg\max_{\theta} \sum_i \log p(\mathbf{s}_i \mid \mathbf{x}_i)$$

over the training data. When training the model, the recurrent dropout [28] is applied to all the biLSTMs. Adam [29] with default settings is used to train the parameters in our experiments. The best iteration is determined by the development performance.

V. EXPERIMENT

A. Tasks

We conduct our experiments on three typical NLP segmentation tasks: chunking, named entity recognition and Chinese word segmentation.

[6]Corresponding numbers of filters are: 32, 32, 32, 16, 8, 8.

TABLE II
STATISTICS OF LABELED DATA

	CoNLL00	CoNLL03	CTB6	PKU	MSR
			number of sentences		
training	8,936	14,987	23,416	17,149	78,232
dev.	1,844	3,466	2,077	1,905	8,692
test	2,012	3,684	2,796	1,944	3,985
			number of tokens		
training	211.7K	204.6K	1,055.5K	1,662.6K	3,633.3K
dev.	44.4K	51.6K	100.3K	163.9K	417.1K
test	47.4K	46.7K	134.1K	172.7K	184.4K

TABLE III
STATISTICS OF UNLABELED DATA

	RCV1	Chinese Gigaword-v5
number of sentences	8.18M	4.82M
number of words	131.22M	473.73M

To evaluate chunking, we use the CoNLL00 text chunking shared task dataset [30], which uses Section 15–18 of Wall Street Journal data (WSJ) as training data and Section 20 as test data. Due to the lack of development data, we use Section 22 of WSJ as the development set and convert constituency trees into chunking data with `chunklink.pl`.[7] To evaluate NER, we use the CoNLL03 dataset [31] and follow the standard training/development/test split. We evaluate CWS on three simplified Chinese datasets: PKU and MSR from 2nd SIGHAN bakeoff and Chinese Treebank 6.0 (CTB6). For the PKU and MSR datasets, the last 10% of the training data are used as development data following Pei *et al.* [32]. For CTB6 data, the recommended data split is used. We convert all the double-byte digits and letters in the PKU data into their single-byte form. All the segmentation performances are evaluated by the F-score.[8] The statistics of the dataset are shown in Table II.

We use the 840B GloVe embeddings released in Pennington *et al.* [33] as fixed word embeddings for English. We use the Word2Vec toolkit released by Ling *et al.* [34] to obtain fixed Chinese character embeddings on Chinese Gigaword Version 5 corpus (Chinese Gigaword-v5).[9]

Unlabeled data are used to train and segment string embeddings. For English data, we use RCV1 as the unlabeled data. For Chinese, we use Chinese Gigaword-v5. The statistics of unlabeled data are shown in Table III.

Reimers and Gurevych [15] pointed out that neural network training is nondeterministic as it usually depends on the seed for the random number generator. To control for this effect, they suggest reporting the average of differently seeded runs. In all our experiments, we set the number of runs to 5.

B. Baseline

We compare our model with the following baselines:

[7][Online]. Available: https://github.com/esrel/DP/blob/master/bin/chunklink.pl
[8]For chunking and NER, we use `conlleval` script to calculate the F-score. For the Chinese word segmentation, the `score` script in 2nd SIGHAN bakeoff is used.
[9][Online]. Available: https://github.com/wlin12/wang2vec

TABLE IV
THE CHUNKING, NER, AND CWS RESULTS OF THE BASELINE MODELS AND OUR NEURAL SEMI-CRF MODELS WITH DIFFERENT INPUT COMPOSITION FUNCTIONS. THE NUMBER AFTER $\pm$ SHOWS THE STANDARD VARIANCE. *SPD.* REPRESENTS THE INFERENCE SPEED AND IS EVALUATED BY THROUGHPUT AGAINST THAT OF NN-LABELER, THUS THE HIGHER, THE BETTER[10]

model	CoNLL00	CoNLL03	CTB6	PKU	MSR	Ave.	spd.
NN-Labeler	$93.31_{\pm 0.14}$	$88.46_{\pm 0.18}$	$93.12_{\pm 0.08}$	$92.87_{\pm 0.10}$	$95.19_{\pm 0.45}$	92.66	1.00x
NN-CRF	$93.84_{\pm 0.09}$	$88.83_{\pm 0.18}$	$93.64_{\pm 0.09}$	$93.57_{\pm 0.04}$	$95.47_{\pm 0.07}$	93.15	0.60x
raw embeddings (Eq. 4)							
SRNN	$93.31_{\pm 0.14}$	$83.37_{\pm 0.29}$	$94.35_{\pm 0.14}$	$94.26_{\pm 0.07}$	$\mathbf{96.51}_{\pm 0.07}$	92.36	0.07x
grConv	$93.07_{\pm 0.12}$	$81.64_{\pm 0.42}$	$94.47_{\pm 0.07}$	$94.24_{\pm 0.13}$	$96.10_{\pm 0.08}$	91.90	0.14x
SAve	$90.15_{\pm 0.24}$	$81.75_{\pm 0.50}$	$91.38_{\pm 0.19}$	$90.61_{\pm 0.08}$	$92.26_{\pm 0.20}$	89.23	0.45x
SCNN	$92.93_{\pm 0.02}$	$86.07_{\pm 0.85}$	$93.90_{\pm 0.07}$	$93.49_{\pm 0.09}$	$95.69_{\pm 0.03}$	92.33	0.08x
SConcat	$92.46_{\pm 0.04}$	$82.41_{\pm 0.77}$	$94.22_{\pm 0.14}$	$94.12_{\pm 0.11}$	$95.78_{\pm 0.05}$	91.80	0.75x
biLSTM (Eq. 3)							
SRNN	$94.25_{\pm 0.06}$	$88.88_{\pm 0.61}$	$94.14_{\pm 0.06}$	$93.91_{\pm 0.11}$	$95.98_{\pm 0.10}$	93.49	0.06x
grConv	$\mathbf{94.43}_{\pm 0.06}$	$89.07_{\pm 0.36}$	$94.43_{\pm 0.09}$	$94.18_{\pm 0.09}$	$95.96_{\pm 0.04}$	**93.66**	0.06x
SAve	$94.11_{\pm 0.20}$	$89.02_{\pm 0.12}$	$93.74_{\pm 0.13}$	$93.53_{\pm 0.08}$	$95.39_{\pm 0.10}$	93.20	0.19x
SCNN	$94.16_{\pm 0.26}$	$\mathbf{89.10}_{\pm 0.61}$	$94.17_{\pm 0.06}$	$94.08_{\pm 0.10}$	$95.74_{\pm 0.08}$	93.45	0.08x
SConcat	$94.31_{\pm 0.07}$	$88.69_{\pm 0.65}$	$\mathbf{94.62}_{\pm 0.05}$	$\mathbf{94.48}_{\pm 0.07}$	$96.11_{\pm 0.07}$	93.64	0.21x
SMinus	$93.87_{\pm 0.17}$	$88.21_{\pm 0.27}$	$93.62_{\pm 0.09}$	$93.55_{\pm 0.10}$	$95.05_{\pm 0.09}$	92.86	0.29x

1) NN-Labeler: The neural network sequence labeling model performing classification on each input observation.
2) NN-CRF: The neural network CRF that models the conditional probability of a label sequence over the input sequence.
3) SRNN: The neural semi-CRF model proposed by Kong *et al.* [10]. The comparison is based on our re-implementation.
4) grConv: The neural semi-CRF model proposed by Zhuo *et al.* [12]. We only adopt their segment representation module (i.e., grConv). The comparison is based on our re-implementation.

BIESO-tag schema is used in all the CRF and sequence labeling models.[11] Both NN-Labeler and NN-CRF take the same contextualized input representation (Eq. 3) as our neural semi-CRF models but vary on the output structure and do not explicitly model segment-level information.

C. Subsequence Encoding

We first study the segment representation by encoding subsequence (§IV-A). This includes a comparison of different *context representing components* and a comparison of different *segment representing components*. The experimental results on chunking, NER, and CWS are shown in Table IV. The first block contains our sequence labeling and CRF baselines. The second block contains the results of semi-CRF models using raw embeddings as input (Eq. 4). The third block contains those using the biLSTM-encoded vector as input (Eq. 3). Among all the compared models in Table IV, grConv with biLSTM context encoding achieves the best test performance, and our SConcat is only 0.02 lower. For each compared task, grConv wins in chunking, SCNN wins in NER. and SConcat wins in two of the three CWS dataset,

By comparing NN-CRF with other semi-CRFs in the third block (these models use the same biLSTM encoded context as input), we can see that the semi-CRFs work better than the linear-chain CRFs in most settings except SMinus. The largest margin between NN-CRF and neural semi-CRF is 0.55 according to the averaged scores. This shows the necessity of modeling the segment in segmentation problems.

The comparison between the second and the third blocks shows that encoding the context is important in neural semi-CRF. Among the compared tasks, there are larger margins[12] between models with and without context encoding on CoNLL00 and CoNLL03, while these margins[13] on CWS are smaller. This observation indicates that chunking and NER are more sensitive to the contextual information of inputs.

In the comparison among different segment representation components, SConcat is only 0.02 points lower than the best grConv baseline. From this comparison, we can also see that properly modeling the input orders is important to the segmentation performance, since the group of networks that models orders (SRNN, grConv, and SConcat) generally works better than the ones that ignore orders (SAve and SMinus).

A further comparison on the inference speed shows that SConcat runs more than 3 times faster than SRNN and grConv, but slower than the NN-Labeler and NN-CRF, which results from the intrinsic difference in time complexity.[14] Considering SConcat's performance and speed advantage, we use SConcat as our composition function in the future experiments.

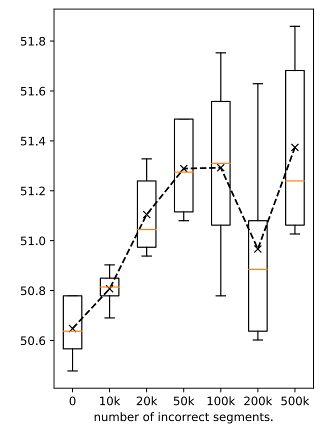
[10]When comparing our results with those in our previous work [1], we found the scores of CoNLL03, CTB6, and PKU are quite close (e.g., the CoNLL03 scores of NN-Labeler is 88.62 and 88.46 between two versions). However, we witness higher MSR results both for the baselines and the neural semi-CRF. This is partially caused by the usage of recurrent dropout and batch training. The multiple-seeded runs also lead to the difference.

[11]O tag which means OUTSIDE is not adopted in CWS experiments since CWS doesn't involve assigning tags to segments.

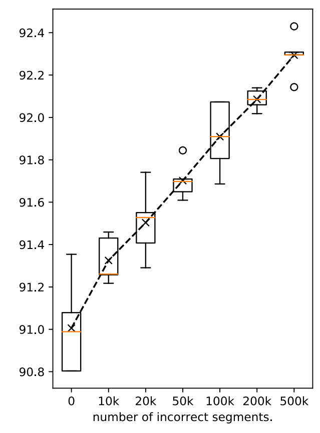
[12]The corresponding numbers are 1.36 and 3.03.

[13]The corresponding numbers are 0.15, 0.22, and -0.40.

[14]The inference speed is evaluated on a single-threaded Intel(R) Xeon(R) CPU E5-2682 v4 without sequence-level parallelization, such as batching.



(a)



(b)

Fig. 4. The relation between number of negative segments and the model's performances. The left Fig. plots that for CoNLL03 and the right Fig. plots that for CTB6.

We need to note that also the results in Table IV can be further improved by the task-specific engineering of the inputs. For example, building the NER models from the character based word representation has been shown to improve the performance [35]. Meanwhile, using the character bigarms as input has also been proved to be effective for CWS compared to the unigrams we use. However, since our goal is to compare the various components in semi-CRFs and compare semi-CRFs with the sequence labeling models, we use a relatively simple and unified inputs.

D. Segment String Embedding

In this section, we study the effect of representing a segment with segment string embeddings by using it as the only representation.

As discussed in Section IV-B, constructing the embedding lookup table might play an important role. We first study the *construction from training data*. We empirically confirm the effect of including incorrect segments in this section. Experiments are conducted on CoNLL03 and CTB6. We build the lookup table on the training data and plot the relation between the number of incorrect segments and the model's performances in Fig. 4. From this figure, more incorrect segments lead to better performances, which indicates the correlation between the number of incorrect segments and the final performance. However, the relatively low performance in Fig. 4 indicates that the *construction from training data* is not optimal. How to effectively construct the lookup table is yet to be answered.

Next, we study the *construction from unlabeled data*, which constructs the lookup table from the automatically segmented unlabeled data. One is the neural semi-CRF (marked as Baseline) baseline which represents segment by composing input The CoNLL03 and CTB6 results of using the lookup table constructed on the unlabeled data are shown in the row marked as "update w/o init." in Table V. It outperforms the models using 500 K incorrect segments in the training data. This shows the effectiveness of *construction from unlabeled data*.

TABLE V
THE EFFECT OF DIFFERENT SEGMENT STRING EMBEDDINGS, INCLUDING THEIR USE AS FIXED, AND UPDATING THEIR WEIGHTS WITH AND WITHOUT INITIALIZATION

model	CoNLL03	CTB6
from training data		
500K update	$51.37_{\pm 0.38}$	$92.13_{\pm 0.11}$
from unlabeled data		
fixed	**74.81**$_{\pm 0.41}$	$91.86_{\pm 0.10}$
update	$73.62_{\pm 0.51}$	**93.65**$_{\pm 0.13}$
update w/o init.	$63.54_{\pm 1.54}$	$92.96_{\pm 0.08}$

TABLE VI
THE RESULTS OF USING BOTH THE COMPOSITION FUNCTION AND SEGMENT STRING EMBEDDINGS. THE FIRST ERROR REDUCTION IS CALCULATED BETWEEN THE MODEL WITH SEGMENT STRING EMBEDDNGS AND NN-CRF. THE SECOND ERROR REDUCTION IS CALCULATED BETWEEN THE MODEL WITH AND WITHOUT SEGMENT STRING EMBEDDNGS

model	CoNLL00	CoNLL03	CTB6	PKU	MSR
dev. result					
NN-CRF	95.26	92.92	94.26	94.55	95.42
biLSTM+SConcat	95.85	93.13	95.30	95.67	96.00
+fixed segment string embeddings					
	95.86	93.52	96.05	96.83	97.07
+update segment string embeddings					
	95.74	**93.77**	**96.22**	**97.03**	**97.45**
test result					
NN-CRF	93.84	88.83	93.64	93.96	95.47
biLSTM+SConcat	94.31	88.69	94.62	94.48	96.11
+segment string embedding according to the dev. result					
	94.39 ± 0.09	**89.87** ± 0.36	**95.63** ± 0.05	**95.67** ± 0.11	**97.25** ± 0.03
Diversity	27.44	10.77	6.08	4.75	3.64
Err. reduction[1]	8.88	9.23	31.17	28.27	39.36
Err. reduction[2]	1.30	10.37	18.61	21.50	29.41

With the auto-segmented data, segment string embeddings are easy to achieve. We use Word2Vec to learn segment string embeddings from the unlabeled data and use them either as fixed input or initialization. The results are also shown in Table V. From this table, adopting these segment embeddings leads to better performance than those that does not. However, using the fixed embeddings may be task-related, since the results on the CoNLL03 dataset favor the fixed segment string embeddings and those on CTB6 perform better using updated word embeddings.

E. Combination

Next, we study the problem of combining the segment representation from subsequence encoding and segment string embeddings. The results are shown in Table VI. From this table, we see that using segment string embeddings leads to a consistent performance improvement. For the NN-CRF baseline, the macro-averaged performance gain of using segment string embeddings is 1.41 and the macro-averaged error reduction is 23.38%, and that for the model without segment string embeddings is 0.92 and 16.24%, respectively. This comparison shows the effectiveness of using segment string embeddings.

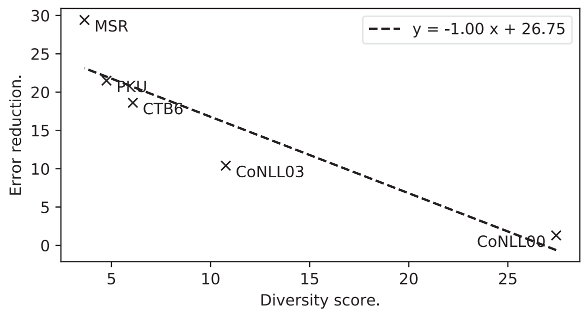


Fig. 5. The relation between diversity and error reduction.

TABLE VII
THE DISTRIBUTION OF TWO TYPES OF ERRORS FOR OUR MODEL WITH AND WITHOUT SEGMENT STRING EMBEDDINGS. THE NUMBERS ARE OBTAINED ON THE RESULTS OF MULTIPLE RUNS ON CONLL03

model	boundary	label
biLSTM+SConcat	354.00	285.60
+segment string embeddings	310.75	269.15
Error reduction	12.25	5.76

From Table VI, we can see that segment string embeddings' effect is task-related, where using segment string embeddings does not significantly change the chunking score but achieves an improvement of more than one point for NER and CWS. We attribute this observation to the fact that chunking presents more diverse segments, which made it difficult to recall enough segments in the segment string embeddings lookup table. We calculate the *segment diversity score* of a dataset as

$$\text{Diversity} = \frac{\text{\# of unique segments}}{\text{\# of segments}}.$$

The diversity score for each dataset is shown in the "Diversity" row of Table VI. CoNLL00 is shown to be the most diverse dataset in our experiments according to this number. A further study shows the correlation between diversity and error reduction with a Pearson correlation of -0.91 (see Fig. 5 for an illustration), and this confirms our assumption.

Since segmentation involves identifying the boundary and assigning labels, we categorize the errors into two types: *boundary errors*, which characterize the incorrect boundaries and *label errors*, which characterize the correct boundary but incorrect labels. We study the effect of segment string embeddings on these two errors. Table VII shows that the segment string embeddings correct more boundary errors than label errors. This observation suggests the possibility to further improve the performance with a better segment string embedding lookup table construction.

A further study on the ways of using segment string embeddings does not reveal significant differences. Chunking favors fixed segment string embeddings slightly more, while the other four tasks favor updating the embeddings during model learning.

F. Using Deep Contextualized Embeddings as Input

ELMo is shown to be very effective in a range of NLP tasks. However, the original study in our previous work predated the rise of contextualized embeddings and leave the question that whether semi-CRF is still useful unanswered. In this paper, we also tried to use ELMo as an input representation (Eq. 2).[15] For English, we use the official release of the large English ELMo trained on the *One Billion Word Benchmark* [36] from Peters *et al.* [13].[16] For Chinese, we use the ELMo toolkit released by Che *et al.* [37] to obtain character unigram ELMo.[17] The character unigram ELMo is trained on Chinese Gigaword-v5. The dimension of ELMo is 1,024 which makes it highly over-parameterized and more prone to overfitting. Thus, we increase the dropout rate to 0.25 in this part of the experiments.

TABLE VIII
THE TEST RESULTS WITH MODEL USING ELMO AS THE INPUT REPRESENTATION. THE FIRST AND SECOND ERROR REDUCTION HAVE THE SAME MEANINGS AS THOSE OF TABLE VI

model	CoNLL00	CoNLL03	CTB6	PKU	MSR
NN-CRF	96.48	91.72	95.91	95.58	96.74
	±0.10	±0.28	±0.05	±0.03	± 0.02
SConcat	96.07	89.31	95.57	94.78	95.30
	±0.12	±0.35	±0.07	±0.11	±0.10
biLSTM+SConcat	**96.60**	91.84	96.11	95.39	96.75
	±0.06	±0.07	±0.12	±0.05	±0.05
+segment string embedding according to dev. results in Table VI					
	96.58	**92.33**	**96.46**	**95.97**	**97.85**
	±0.09	±0.05	±0.13	±0.08	±0.06
Diversity	27.44	10.77	6.08	4.75	3.64
Err. reduction[1]	2.73	7.36	13.23	8.94	33.92
Err. reduction[2]	-0.67	5.91	8.81	13.12	33.81

The experimental results are shown in Table VIII. These results confirm the trend we witnessed in Table VI, in which our semi-CRF with SConcat outperforms the neural CRF model in three out of five datasets, and incorporating segment string embeddings further increases the model's performance in four out of the five datasets. The correlation between segment diversity and error reduction is consistent with that of Fig. 5. The difference between the model with and without segment string embeddings is not significant on chunking, which might result from the diverse segment forms in the CoNLL00 dataset.

The results of additional experiments in which our model directly uses ELMo as input is shown in the "SConcat" row. The segmentation performance is improved against those in Table IV. We attribute this to the fact that ELMo models context information. However, the score lags that of using biLSTM to encode context (biLSTM+SConcat), which shows the necessity of modeling task-specific context.

From Table VIII, the macro-averaged error reduction achieved by using segment string embeddings against the NN-CRF baseline is 13.16 and this number is 12.00 against the neural semi-CRF model without segment string embeddings, yielding a corresponding macro-averaged performance gains are 0.55 and 0.49, respectively. From this result, the reduction is smaller compared with that of Table VI. In addition, we can see that the effect of subsequence encoding and segment string embeddings

[15]ELMo is used as a replacement for $\phi^{(\text{fix})} \oplus \phi^{(\text{tune})}$ in Eq. 9.
[16][Online]. Available: https://allennlp.org/elmo
[17][Online]. Available: https://github.com/HIT-SCIR/ELMoForManyLangs/

TABLE IX
COMPARISON WITH THE SOTA CHUNKING SYSTEMS. $\clubsuit$ MARKS THE SYSTEM THAT USES MULTI-TASK LEARNING. $\heartsuit$ MARKS THE SYSTEM THAT USES SEMI-SUPERVISED LEARNING. $*$ MARKS THE RESULT THAT IS NOT DIRECTLY COMPARABLE DUE TO THE DATA SPLIT DIFFERENCE. $\diamondsuit$ MARKS THE RESULT THAT IS OBTAINED FROM THE AVERAGE OF MULTIPLE RUNS

model	CoNLL00
CVT+Multi-task (Large)$^{\heartsuit\clubsuit\diamondsuit}$ [37]	**97.0**
Flair embeddings$^{\heartsuit\diamondsuit}$ [38]	96.72
ours$^{\heartsuit\diamondsuit}$	96.58
TagLM$^{*\heartsuit\diamondsuit}$ [39]	96.37
LM-LSTM-CRF$^{*\heartsuit\clubsuit\diamondsuit}$ [40]	95.96
JMT$^{\clubsuit}$ [41]	95.77
Low supervision [42]	95.57
Suzuki and Isozaki (2008)$^{\heartsuit}$ [43]	95.15
grConv (reported)$^{\heartsuit}$ [12]	95.10
NCRF++ [44]	95.06

TABLE X
COMPARISON WITH THE SOTA NER SYSTEMS. $\clubsuit$, $\heartsuit$, AND $\diamondsuit$ HAVE THE SAME MEANINGS AS THOSE IN TABLE IX. $*$ MARKS THE RESULT THAT IS NOT DIRECTLY COMPARABLE DUE TO USING TRAIN AND DEVELOPMENT SPLITS FOR TRAINING

model	CoNLL03
CNN Large+fine-tune$^{\heartsuit}$ [45]	**93.5**
Flair embeddings$^{*\heartsuit\diamondsuit}$ [38]	93.09
BERT Large$^{\heartsuit}$ [46]	92.8
CVT+Multi-task (Large)$^{\clubsuit\heartsuit\diamondsuit}$ [37]	92.6
ours$^{\heartsuit\diamondsuit}$	92.33
BiLSTM-CRF+ELMo$^{\heartsuit\diamondsuit}$ [13]	92.22
TagLM$^{*\heartsuit\diamondsuit}$ [39]	91.93
HSCRF$^{\heartsuit\diamondsuit}$ [47]	91.38
NCRF++ [44]	91.35
LM-LSTM-CRF$^{\heartsuit\clubsuit\diamondsuit}$ [40]	91.24
BiLSTM-CRF-CNN [14]	91.21
LSTM-CRF [34]	90.94
grConv (reported)$^{\heartsuit}$ [12]	90.87

is not fully orthogonal. With the enhanced input representation, the benefits from using segment string embedding are decreased. However, the fact that segment embeddings still lead to improvement shows the effectiveness of our method.

Utill this point in our study, we have empirically studied different context representations, different composition functions, and different segment string embeddings. Therefore, from the experimental results, the following knowledge about our neural semi-CRF model is obtained:

- Neural semi-CRFs outperform the linear-chain CRFs in most of the cases (see Table IV and VIII).
- Encoding context is important, even when the input representation carries contextual information (see Table IV and VIII).
- Using segment string embeddings learned on unlabeled data helps the tasks where the segment diversity is not too large (see Table V, VI, and VIII).

G. Comparison With State-of-the-Art Systems

Finally, we compare our neural semi-CRF model with the SOTA segmentation systems. Table IX shows the chunking comparison, where our neural semi-CRFs using ELMo as input outperforms most of the compared systems. It only lags the SOTA system trained with multi-task learning [38] by 0.44, and lags the system that uses rich character-level and word-level contextualized embeddings [39] by 0.14. The comparison with Akbik *et al.* [39] again emphasizes the importance of the input representation. In addition, we need to note that the dependency parsing training data (Section 2-21 in WSJ) in Clark *et al.* [38] overlaps with the test set of CoNLL00 (Section 20 in WSJ). Chunking and dependency parsing is syntactically correlated; therefore, their results not very convincing. We also attribute the gap comparing our model with that of Akbik *et al.* [39] to their learning of contextualized embeddings on both the word and character level.

Table X shows the NER comparison. The trend is similar to that of Table IX, in which improving the input representation [18], [39] and training with multi-task learning [38] helps NER considerably. In Table X, our system is 1.17 points lower than the SOTA results of cloze-driven self-attention network [46]. We attribute this to the fact that cloze-driven self-attention network was trained on a larger corpus (3,300 M vs 800 M words for ELMo) and carefully tailored language modeling scheme. However, our model outperforms the system that carefully engineers the usage of ELMo in biLSTM-CRF [13], [40], which shows the effectiveness of modeling the composition and representing the segment with embeddings in the segmentation problem.

TABLE XI
COMPARISON WITH THE SOTA CWS SYSTEMS. † MARKS THE SYSTEM THAT USES THE CHARACTER UNIGRAM AS INPUT. ‡ MARKS THE SYSTEM THAT USES THE CHARACTER BIGRAM AS INPUT. $\heartsuit$ AND $\diamondsuit$ HAVE THE SAME MEANINGS AS THOSE IN TABLE IX. $*$ MARKS THE RESULT THAT IS NOT DIRECTLY COMPARABLE DUE TO PREPROCESSING ON DIGITS AND ENGLISH LETTERS ACCORDING TO PENG AND DREDZE [48]

model	CTB6	PKU	MSR
BiLSTM-CRF+hyper-param. search†‡ [49]	**96.7**	96.1	**98.1**
ours$^{\dagger\heartsuit\diamondsuit}$	96.5	96.0	97.9
Transition-based+emb. tuning†‡ [50]	96.2	**96.3**	97.5
Greedy Search+word context† [23]	96.2	96.0	97.8
BiLSTM-CRF+adv. loss†‡ [51]		94.3	96.0
Greedy Search+Span repr.† [52]	-	95.8	97.1
Greedy Search† [19]	-	95.7	96.4
Gated Recursive NN*†‡ [53]	95.8	96.4	97.6
LSTM*†‡ [54]	94.9	95.7	96.4
Max-margin Tensor NN†‡ [31]	-	95.2	97.2
CNN† [55]	-	92.4	93.3
SRNN (reported)† [10]	-	90.6	90.7
Sparse semi-CRF [7]	-	95.2	97.3
Sparse transition-based [56]	-	95.1	97.2
Auto-segmented Features$^{\heartsuit}$ [25]	95.7	-	-

Table XI shows the comparison with the SOTA CWS systems. From this table, we can see that using the character bigrams as input is key to SOTA performance in previous CWS works. Our neural semi-CRF model achieves competitive performance with these systems by only using the character unigrams as input. On CTB6, our model's performance is only 0.24 lower than the SOTA biLSTM-CRF system with the careful hyperparameter search [49]. On PKU, the gap is 0.33 and 0.25 for MSR. We need to note that all the works in Table XI report a single score and this practice can be problematic according to Reimers and Gurevych [15]. Our results were obtained from the average of multiple runs which is more reliable.

The overall gap between the results of our model and the aggregation of the results of the SOTA segmentation systems is 0.43. Our model can be concluded as competitive to the SOTA system according to this comparison, which once again shows the effectiveness of representing the segment with neural semi-CRFs both by encoding subsequence and embedding the segment string.

VI. Related Work

Semi-CRFs have been successfully used in many NLP tasks such as information extraction [9], named entity recognition [3], opinion extraction [4], disfluency detection [5], and Chinese word segmentation [6], [7]. There are also plenty applications of semi-CRFs to the other areas, such as speech [57]–[59]. However, most of the NLP works use sparse features as input. Its combination with a neural network representation module is relatively less studied. We based our work on the neural semi-CRF work of Kong *et al.* [10] and Zhuo *et al.* [12], and dervised a considerable amount of meaningful extensions. In addition, a thorough study on the model structure revealed the important component in neural semi-CRFs.

Sequence labeling is a widely used proxy for segmentation. High accuracy of segmentation is achieved by improving the input representation [13], [18], [39]–[41], and using a relatively simple structure prediction model such as classification. We observed boosted performance with semi-CRFs that directly models the segment using the powerful embeddings from ELMo as input. This suggests that modeling structure is still beneficial even in the condition of pushing the input representation to the extreme.

Using auto-segmented data to enhance Chinese word segmentation has been studied in Wang *et al.* [26]. However, only statistical features that are counted on the auto-segmented data were introduced to help to determine segment boundary and the entire segment was not considered in their work. Our model takes the advantage of the word embedding technique [17] and explicitly represents the entire segment.

In this paper, we apply our model to three segmentation problems, including chunking, NER, and CWS. Many other NLP problems can be modeled as segmentation such as disfluency detection [5], information extraction, and slot filling [60] in spoken language understanding. It would be interesting to evaluate the application of our neural semi-CRF model on these problems.

VII. Conclusion

In this paper, we thoroughly study the problem of representing a segment in neural semi-CRF model. We empirically test the importance of context representation in neural semi-CRFs. We propose a concatenation alternative for segment representation that achieves an equivalent accuracy to SRNN and grConv but runs faster. We also propose an effective way of incorporating segment string embeddings as the segment representation and find that it significantly improves the performance. Experiments on chunking, NER, and CWS show that the neural semi-CRFs benefits from the rich segment representation and achieves competitive performance with the state-of-the-art systems.

References

[1] Y. Liu, W. Che, J. Guo, B. Qin, and T. Liu, "Exploring segment representations for neural segmentation models," 2016, [Online]. Available: http://arxiv.org/abs/1604.05499

[2] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *Proc. Human Lang. Technol. Conf. North Amer. Chapter Assoc. Comput. Linguist.*, 2003, pp. 213–220. [Online]. Available: https://www.aclweb.org/anthology/N03-1028

[3] D. Okanohara, Y. Miyao, Y. Tsuruoka, and J. Tsujii, "Improving the scalability of semi-Markov conditional random fields for named entity recognition," in *Proc. 21st Int. Conf. Computat. Linguist. 44th Annu. Meeting Assoc. Computat. Linguist.*, Jul. 2006, pp. 465–472.

[4] B. Yang and C. Cardie, "Extracting opinion expressions with semi-Markov conditional random fields," in *Proc. Joint Conf. Empir. Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, Jeju Island, Korea, Jul. 2012, pp. 1335–1345.

[5] J. Ferguson, G. Durrett, and D. Klein, "Disfluency detection with a semi-Markov model and prosodic features," in *Proc. Annu. Conf. North Amer. Chapter Assoc. Comput. Linguist.*, 2015, pp. 257–262. [Online]. Available: http://www.aclweb.org/anthology/N15-1029

[6] G. Andrew, "A hybrid Markov/semi-Markov conditional random field for sequence segmentation," in *Proc. Empir. Methods Natural Lang. Process.*, 2006, pp. 465–472.

[7] X. Sun, Y. Zhang, T. Matsuzaki, Y. Tsuruoka, and J. Tsujii, "A discriminative latent variable Chinese segmenter with hybrid word/character information," in *Proc. Human Lang. Technol.: Annu. Conf. North Amer. Chapter Assoc. Comput. Linguist.*, Jun. 2009, pp. 56–64.

[8] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Mach. Learn.*, Jun. 2001, pp. 282–289.

[9] S. Sarawagi and W. W. Cohen, "Semi-Markov conditional random fields for information extraction," *Adv. Neural Inf. Process. Syst.*, vol. 17, 2004, pp. 1185–1192.

[10] L. Kong, C. Dyer, and N. A. Smith, "Segmental recurrent neural networks," in *Proc. 4th Int. Conf. Learn. Repr.*, May 2016.

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

[12] J. Zhuo, Y. Cao, J. Zhu, B. Zhang, and Z. Nie, "Segment-level sequence modeling using gated recursive semi-Markov conditional random fields," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist.*, 2016, pp. 1413–1423.

[13] M. Peters *et al.*, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol.*, Jun. 2018, pp. 2227–2237.

[14] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist*, 2016, pp. 1064–1074.

[15] N. Reimers and I. Gurevych, "Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging," in *Proc. Empir. Methods Natural Lang. Process.*, 2017, pp. 338–348.

[16] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguist.*, vol. 5, no. 1, pp. 135–146, 2017.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2013, vol. 2, pp. 3111–3119.

[18] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol.*, Jun. 2019, pp. 4171–4186.

[19] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proc. 52th Annu. Meeting Assoc. Comput. Linguist.*, 2014, pp. 655–665. [Online]. Available: http://www.aclweb.org/anthology/P14-1062

[20] D. Cai and H. Zhao, "Neural word segmentation learning for chinese," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist.*, 2016, pp. 409–420. [Online]. Available: http://www.aclweb.org/anthology/P16-1039

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

[22] W. Wang and B. Chang, "Graph-based dependency parsing with bidirectional LSTM," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist.*, 2016, pp. 2306–2315. [Online]. Available: http://www.aclweb.org/anthology/P16-1218

[23] J. Cross and L. Huang, "Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles," in *Proc. Empir. Methods Natural Lang. Process.*, 2016, pp. 1–11. [Online]. Available: https://aclweb.org/anthology/D16-1001

[24] H. Zhou, Z. Yu, Y. Zhang, S. Huang, X.-Y. Dai, and J. Chen, "Word-context character embeddings for chinese word segmentation," in *Proc. Empir. Methods Natural Lang. Process.*, 2017, pp. 760–766. [Online]. Available: https://www.aclweb.org/anthology/D17-1079

[25] W. Chen, J. Kazama, K. Uchimoto, and K. Torisawa, "Improving dependency parsing with subtrees from auto-parsed data," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, Aug. 2009, pp. 570–579.

[26] 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 *Proc. 5th Int. Joint Conf. Natural Lang. Process.*, Nov. 2011, pp. 309–317.

[27] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," in *Proc. 53th Annu. Meeting Assoc. Comput. Linguist.*, 2015, pp. 334–343.

[28] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Proc. 30th Int. Conf. Neural Inform. Process. Syst.*, Barcelona, Spain, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?id=3157096.3157211

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

[30] E. F. T. K. Sang and S. Buchholz, "Introduction to the conll-2000 shared task: Chunking," in *Proc. Conf. Comput. Natural Lang. Learn.*, 2000. [Online]. Available: https://doi.org/10.3115/1117601.1117631

[31] E. F. T. K. Sang and F. De Meulder, "Introduction to the coNLL-2003 shared task: Language-independent named entity recognition," in *Proc. Seventh Conf. Natural Lang. Learn. HLT-NAACL*, 2003, pp. 142–147.

[32] W. Pei, T. Ge, and B. Chang, "Max-margin tensor neural network for Chinese word segmentation," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguist.*, Jun. 2014, pp. 293–303.

[33] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, Oct. 2014, pp. 1532–1543.

[34] W. Ling, C. Dyer, A. W. Black, and I. Trancoso, "Two/too simple adaptations of word2vec for syntax problems," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol.*, May/Jun. 2015, pp. 1299–1304.

[35] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol.*, Jun. 2016, pp. 260–270.

[36] C. Chelba *et al.*, "One billion word benchmark for measuring progress in statistical language modeling," in *Proc. Interspeech 15th Annu. Conf. Int. Speech Commun. Assoc.*, Singapore, Sep. 2014. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2014/i14_2635.html

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

[38] K. Clark, M.-T. Luong, C. D. Manning, and Q. Le, "Semi-supervised sequence modeling with cross-view training," in *Proc. Empir. Methods Natural Lang. Process.*, 2018, pp. 1914–1925. [Online]. Available: http://www.aclweb.org/anthology/D18-1217

[39] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proc. Coling*, 2018, pp. 1638–1649. [Online]. Available: http://www.aclweb.org/anthology/C18-1139

[40] M. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguist.*, 2017, pp. 1756–1765. [Online]. Available: http://aclweb.org/anthology/P17-1161

[41] L. Liu *et al.*, "Empower sequence labeling with task-aware neural language model," in *Proc. Assoc. Adv. Artif. Intell.*, 2018.

[42] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint many-task model: Growing a neural network for multiple NLP tasks," in *Proc. Empir. Methods Natural Lang. Process.*, 2017, pp. 1923–1933. [Online]. Available: https://www.aclweb.org/anthology/D17-1206

[43] A. Søgaard and Y. Goldberg, "Deep multi-task learning with low level tasks supervised at lower layers," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist.*, 2016, pp. 231–235. [Online]. Available: http://anthology.aclweb.org/P16-2038

[44] J. Suzuki and H. Isozaki, "Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data," in *Proc. ACL-08: HLT,* Columbus, Ohio, USA, Jun. 2008, pp. 665–673,.

[45] J. Yang and Y. Zhang, "Ncrf++: An open-source neural sequence labeling toolkit," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguist.*, 2018, pp. 74–79. [Online]. Available: http://www.aclweb.org/anthology/P18-4013

[46] A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, and M. Auli, "Cloze-driven pretraining of self-attention networks," 2019. [Online]. Available: http://arxiv.org/abs/1903.07785

[47] Z. Ye and Z.-H. Ling, "Hybrid semi-markov CRF for neural sequence labeling," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguist.*, 2018, pp. 235–240. [Online]. Available: http://www.aclweb.org/anthology/P18-2038

[48] N. Peng and M. Dredze, "Improving named entity recognition for chinese social media with word segmentation representation learning," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist.*, 2016, pp. 149–155. [Online]. Available: http://anthology.aclweb.org/P16-2025

[49] J. Ma, K. Ganchev, and D. Weiss, "State-of-the-art Chinese word segmentation with Bi-LSTMs," in *Proc. Empir. Methods Natural Lang. Process.*, 2018, pp. 4902–4908. [Online]. Available: http://www.aclweb.org/anthology/D18-1529

[50] J. Yang, Y. Zhang, and F. Dong, "Neural word segmentation with rich pretraining," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguist.*, 2017, pp. 839–849. [Online]. Available: http://aclweb.org/anthology/P17-1078

[51] X. Chen, Z. Shi, X. Qiu, and X. Huang, "Adversarial multi-criteria learning for Chinese word segmentation," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguist.*, 2017, pp. 1193–1203. [Online]. Available: http://aclweb.org/anthology/P17-1110

[52] D. Cai, H. Zhao, Z. Zhang, Y. Xin, Y. Wu, and F. Huang, "Fast and accurate neural word segmentation for chinese," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguist.*, 2017, pp. 608–615. [Online]. Available: http://aclweb.org/anthology/P17-2096

[53] X. Chen, X. Qiu, C. Zhu, and X. Huang, "Gated recursive neural network for chinese word segmentation," in *Proc. 53th Annu. Meeting Assoc. Comput. Linguist.*, 2015, pp. 1744–1753. [Online]. Available: http://www.aclweb.org/anthology/P15-1168

[54] X. Chen, X. Qiu, C. Zhu, P. Liu, and X. Huang, "Long short-term memory neural networks for Chinese word segmentation," in *Proc. Empirical Methods Natural Lang. Process.*, 2015, pp. 1197–1206. [Online]. Available: http://aclweb.org/anthology/D15-1141

[55] X. Zheng, H. Chen, and T. Xu, "Deep learning for Chinese word segmentation and POS tagging," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, Oct. 2013, pp. 647–657.

[56] Y. Zhang and S. Clark, "Chinese segmentation with a word-based perceptron algorithm," in *Proc. 45th Annu. Meeting Assoc. Comput. Linguist.*, Jun. 2007, pp. 840–847.

[57] G. Zweig and P. Nguyen, "A segmental CRF approach to large vocabulary continuous speech recognition," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, Merano, 2009, pp. 152–157.

[58] Y. He and E. Fosler-Lussier, "Efficient segmental conditional random fields for one-pass phone recognition," in *Proc. Interspeech, 13th Annu. Conf. Int. Speech Commun. Assoc.*, Portland, Oregon, USA, Sep. 2012, pp. 1898–1901.

[59] H. Tang *et al.*, "End-to-end neural segmental models for speech recognition," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 8, pp. 1254–1264, Dec. 2017.

[60] G. Mesnil *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 3, pp. 530–539, Mar. 2015.