Neural Sequence Segmentation as Determining the Leftmost Segments

Yangming Li[1], **Lemao Liu**[1], **Kaisheng Yao**[2]
[1]Tencent AI Lab
[2]Ant Group
{newmanli,redmondliu}@tencent.com
kaisheng.yao@antgroup.com

Abstract

Prior methods to text segmentation are mostly at token level. Despite the adequacy, this nature limits their full potential to capture the long-term dependencies among segments. In this work, we propose a novel framework that incrementally segments natural language sentences at segment level. For every step in segmentation, it recognizes the leftmost segment of the remaining sequence. Implementations involve LSTM-minus technique to construct the phrase representations and recurrent neural networks (RNN) to model the iterations of determining the leftmost segments. We have conducted extensive experiments on syntactic chunking and Chinese part-of-speech (POS) tagging across 3 datasets, demonstrating that our methods have significantly outperformed previous all baselines and achieved new state-of-the-art results. Moreover, qualitative analysis and the study on segmenting long-length sentences verify its effectiveness in modeling long-term dependencies.

Sentence	Tangible capital will be about $ 115 million .
Segments	(Tangible capital, NP), (will be, VP), (about $ 115 million, NP), (., O)
IOB Tags	B-NP I-NP B-VP I-VP B-NP I-NP I-NP I-NP O
Transition Actions	SHIFT SHIFT REDUCE-NP SHIFT SHIFT REDUCE-VP SHIFT SHIFT SHIFT SHIFT REDUCE-NP OUT

Table 1: The first two rows show an example extracted from CoNLL-2000 dataset (Sang and Buchholz, 2000). The last two rows are two types of token-level labels commonly used to represent the segments.

1 Introduction

Sequence segmentation, as an important task in natural language understanding (NLU), partitions a sentence into multiple segments. The first two rows of Table 1 show a case from a syntactic chunking dataset. The input sentence is a sequence of tokens and the output segments are multiple labeled phrases. These segments are nonoverlapping and fully cover the input sentence.

In previous works, there are two dominant approaches to sequence segmentation. The most common is to regard it as a sequence labeling problem with resorting to IOB tagging scheme (Huang et al., 2015; Akbik et al., 2018; Liu et al., 2019). This method is simple yet very effective, providing tons of state-of-the-art performances. For example, Huang et al. (2015) present Bidirectional LSTM-CRF for named entity recognition (NER) and POS tagging, which adopts BiLSTM (Hochreiter and Schmidhuber, 1997) to read the input sentence and CRF (Lafferty et al., 2001) to decode the label sequence. An alternative method employs a transition-based system to incrementally segment and label an input token sequence (Zhang et al., 2016, 2018). For instance, Zhang et al. (2016) present a transition-based model for Chinese word segmentation that exploits not only character embedding but also token embedding. This type of method enjoys a number of attractive properties, including theoretically lower time complexity and capturing non-local features.

The above two approaches are essentially at token level, where a single segment is represented by multiple token-level labels (e.g., transition actions). In spite of the adequacy, the labels used to model the relation among output segments are far more than the segments themselves. As demonstrated in Figure 1, modeling the transition between the two segments, “will be" and “about $ 115 million", consumes 6 IOB tags or 8 transition actions. This ill-posed design certainly limits the full potential of segmentation models to capture the long-term dependencies among segments.

Previously, Kong et al. (2015) attempted to de-

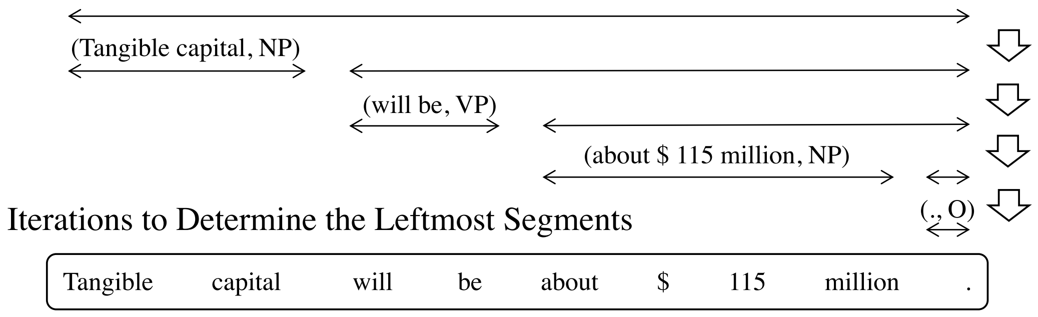
(Tangible capital, NP)
(will be, VP)
(about $ 115 million, NP)
(., O)
Iterations to Determine the Leftmost Segments
Tangible capital will be about $ 115 million .

Figure 1: This case illustrates how our method segments a natural language sentence.

velop segment-level models, which define a joint probability distribution over the partition of an input sentence and the labeling of the segments. Such an approach circumvents using token-level labels. However, we find that, in experiments, it underperforms current token-level models. Moreover, its use of dynamic programming (DP) incurs quadratic running time, which is too slow for both training and inference (see Section 3.7).

In this paper, we introduce a novel framework that incrementally segments a sentence at segment level. The segmentation process is iterative and incremental. At each iteration, the proposed framework determines the leftmost segment of the remaining unprocessed sequence. Under this scheme, we don't resort to token-level labels and enjoy linear time complexity. The implementation contains two stages. Firstly, we utilize LSTM-minus (Wang and Chang, 2016; Cross and Huang, 2016) technique to construct the representations for all the phrases. Secondly, we adopt LSTM to model the iterative segmentation process, which captures the strong correlation among segments. At every step, the input consists of the previous segment and the remaining unprocessed sequence, and the output is the leftmost segment.

Figure 1 depicts how our framework segments the sentence in Table 1. The output segments are obtained in an iterative and incremental manner. At each iteration, the leftmost segment of the remaining sequence is extracted and labeled. Compared with token-level models, we take much fewer steps to complete the segmentation process.

Extensive experiments have been conducted on syntactic chunking and Chinese part-of-speech (POS) tagging across 3 datasets. The proposed framework has obtained new state-of-the-art performances on all of them. Besides, qualitative study and the results on segmenting long-length sentences confirm its effectiveness in capturing long-term dependencies.

Our contributions are as follows:

- we present a novel framework that incrementally segments a natural language sentence at segment level. In comparison, previous approaches are mostly at token-level;
- we have notably outperformed previous baselines and established new state-of-the-art results on the 3 datasets of syntactic chunking and Chinese POS tagging. Experiments also show that our model is competitive with strong NER baselines;
- compared with prior methods, our model well captures the long-term dependencies among segments. This is strongly verified by qualitative study and the experiment on segmenting long-length sentences.

The source code of this work is available at https://github.com/LeePleased/LeftmostSeg.

2 Architecture

We denote a n-length input sentence as $\mathbf{x} = [x_1, x_2, \cdots, x_n]$, where x_i is a token (such as a word or a character). The output segments are represented as $\mathbf{y} = [y_1, y_2, \cdots, y_m]$, where m is the amount of segments. Every segment y_k is denoted as a triple (i_k, j_k, l_k). (i_k, j_k) is the span of the segment, corresponding to the phrase $\mathbf{x}_{i_k,j_k} = [\mathbf{x}_{i_k}, \mathbf{x}_{i_k+1}, \cdots, \mathbf{x}_{j_k}]$. l_k is from a predefined label space $\mathcal{L}$ and specifies the label of the segment. These segments are non-overlapping and fully cover the input sentence.

The example in Table 1 is represented as

$$\begin{aligned} \mathbf{x} &= [\text{Tangible}, \text{capital}, \text{will}, \text{be}, \text{about}, \$, 115, \\ &\quad \text{million}, .] \\ \mathbf{y} &= [(1,2,\text{NP}), (3,4,\text{VP}), (5,8,\text{NP}), (9,9,\text{O})] \end{aligned} .$$

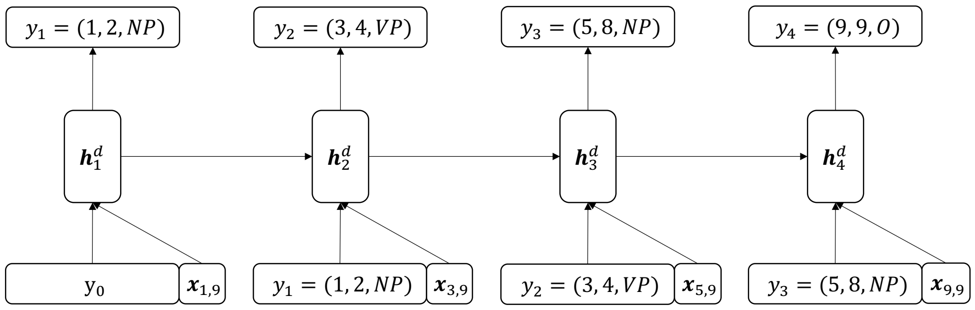

Figure 2: An example to show the incremental process to recognize leftmost segments.

2.1 Phrase Representation Construction

Our goal here is to construct the representation for every phrase $\mathbf{x}_{i,j}$. Later, we will use the phrase representation to embed the segment y_{k-1} and the unprocessed sequence $\mathbf{x}_{i_k,n}$.

Firstly, each token x_k is represented as

$$\mathbf{e}_k = \mathbf{E}^{\text{t}}(x_k) \oplus \text{CharCNN}(x_k), \tag{1}$$

where $\mathbf{E}^{\text{t}}$ is a token embedding matrix and $\oplus$ is the column-wise vector concatenation. Following previous works (Ma and Hovy, 2016; Liu et al., 2019), we use CharCNN to extract the character-level representations.

Secondly, we utilize bidirectional LSTMs $\overrightarrow{f}^{\text{e}}$ and $\overleftarrow{f}^{\text{e}}$ to compute the context-sensitive representation for each token x_k:

$$\begin{cases} \overrightarrow{\mathbf{h}}^{\text{c}}_k = \overrightarrow{f}^{\text{e}}(\overrightarrow{\mathbf{h}}^{\text{c}}_{k-1}, \mathbf{e}_k) \\ \overleftarrow{\mathbf{h}}^{\text{c}}_k = \overleftarrow{f}^{\text{e}}(\overleftarrow{\mathbf{h}}^{\text{c}}_{k+1}, \mathbf{e}_k) \\ \mathbf{h}^{\text{c}}_k = \overrightarrow{\mathbf{h}}^{\text{c}}_k \oplus \overleftarrow{\mathbf{h}}^{\text{c}}_k \end{cases}. \tag{2}$$

Inspired by previous works (Wang and Chang, 2016; Gaddy et al., 2018) in syntactic analysis, we integrate LSTM-minus features into phrase representations. Specifically, the representation $\mathbf{h}^{\text{p}}_{i,j}$ for a phrase $\mathbf{x}_{i,j}$ is computed as the concatenation of the difference of LSTM hidden states:

$$\mathbf{h}^{\text{p}}_{i,j} = \mathbf{h}^{\text{c}}_j \oplus (\mathbf{h}^{\text{c}}_j - \mathbf{h}^{\text{c}}_i) \oplus \mathbf{h}^{\text{c}}_i. \tag{3}$$

Inside algorithm (Lari and Young, 1990) is another method to extracting phrase representation. Its advantage is to incorporate the potential hierarchical structure of natural language without using treebank annotation. For example, Drozdov et al. (2019) utilize inside algorithm to recursively compute the content representations. Despite the attractive property, its time complexity $\mathcal{O}(n^3)$ is too inefficient to practice use.

2.2 Leftmost Segment Determination

Figure 2 demonstrates how our method iteratively and incrementally segments the sentence in Table 1. LSTM is used as the backbone to model the iterative process. At every step, the input consists of the prior segment and the unprocessed sequence, and the output is the predicted segment.

Firstly, we embed the previous segment y_{k-1} and the unprocessed sequence $\mathbf{x}_{i_k,n}$. The previous segment is represented as

$$\mathbf{h}^{\text{s}}_{k-1} = \begin{cases} \mathbf{h}^{\text{p}}_{i_{k-1},j_{k-1}} \oplus \mathbf{E}^{\text{l}}(l_{k-1}) & k > 1 \\ \mathbf{v} & k = 1 \end{cases}, \tag{4}$$

where $\mathbf{E}^{\text{l}}$ is a label embedding matrix and $\mathbf{v}$ is a trainable vector. The unprocessed sequence is embedded as $\mathbf{h}^{\text{p}}_{i_k,n}$.

At each iteration k, we use another LSTM f^{d} to model the dependency among segments:

$$\mathbf{h}^{\text{d}}_k = f^{\text{d}}(\mathbf{h}^{\text{d}}_{k-1}, \mathbf{h}^{\text{s}}_{k-1} \oplus \mathbf{h}^{\text{p}}_{i_k,n}). \tag{5}$$

During training, i_k is known since the ground truth segments $\mathbf{y}$ is reachable. At evaluation time, we set $i_k = j_{k-1} + 1$.

Then, we separately predict the span and the label of a segment. We define a set $\mathcal{S}_k$ containing all valid span candidates for prediction:

$$\mathcal{S}_k = \{(i_k, i_k), (i_k, i_k + 1), \cdots, (i_k, n)\}. \tag{6}$$

The probability of a span $(i, j) \in \mathcal{S}_k$ is

$$Q^{\text{s}}_{k,i,j} \propto \exp\big((\mathbf{h}^{\text{d}}_k)^\top \mathbf{W}^{\text{s}} \mathbf{h}^{\text{p}}_{i,j}\big). \tag{7}$$

The probability of a label $l \in \mathcal{L}$ for a span (i, j) is

$$Q^{\text{l}}_{k,i,j,l} \propto \exp\big(\mathbf{E}^{\text{l}}(l)^\top \mathbf{W}^{\text{l}}(\mathbf{h}^{\text{p}}_{i,j} \oplus \mathbf{h}^{\text{d}}_k)\big). \tag{8}$$

The matrices $\mathbf{W}^{\text{s}}$ and $\mathbf{W}^{\text{l}}$ are learnable.

```
Algorithm 1: Inference Procedure
Input: The representations for all the phrases, $\mathbf{h}^{\mathrm{p}}_{i,j}, 1 \le i \le j \le n$.
Output: The sequence of predicted segments, $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{\hat{m}}]$.
1  Set a list $\mathbf{y}$ as [] and set a counter as $k = 1$.
2  Denote the remaining input token list as $\mathbf{x}$.
3  Initialize the representation for previous segment as $\mathbf{v}$.
4  Initialize the representation for unprocessed sequence as $\mathbf{h}^{\mathrm{p}}_{1,n}$.
5  while $\mathbf{x}$ is not empty do
6      Get LSTM hidden state $\mathbf{h}^{\mathrm{d}}_k$ by using Equation 5.
7      Predict a segment $\hat{y}_k$ by using the Equations from 7 to 9.
8      Append the new segment $\hat{y}_k$ into list $\hat{\mathbf{y}}$.
9      Reset the representation for previous segment as $\mathbf{h}^{\mathrm{p}}_{\hat{i}_k,\hat{j}_k} \oplus \mathbf{E}^{\mathrm{l}}(\hat{l}_k)$.
10     Reset the representation for unprocessed sequence as $\mathbf{h}^{\mathrm{p}}_{\hat{j}_k+1,n}$.
11     Pop the tokens $[x_{\hat{i}_k}, x_{\hat{i}_k+1}, \cdots, x_{\hat{j}_k}]$ from the remaining tokens $\mathbf{x}$.
12     Increase the counter by 1: $k = k + 1$.
13 The amount of predicted segments: $\hat{m} = k$.
```

Finally, the leftmost segment is obtained as

$$\begin{cases} (\hat{i}_k, \hat{j}_k) = \underset{(i,j)\in\mathcal{S}_k}{\arg\max}\, Q^{\mathrm{s}}_{k,i,j} \\ \hat{l}_k = \underset{l\in\mathcal{L}}{\arg\max}\, Q^{\mathrm{l}}_{k,i,j,l} \\ \hat{y}_k = (\hat{i}_k, \hat{j}_k, \hat{l}_k) \end{cases} \cdot \tag{9}$$

The iterative process ends when the remaining sequence $\mathbf{x}_{j_k+1,n}$ is empty (i.e., $j_k = n$).

2.3 Training and Inference

During training, we use teacher forcing where every segment y_k is predicted using its previous ground-truth segments $[y_1, y_2, \cdots, y_{k-1}]$. A hybrid loss is induced as

$$\mathcal{J} = -\sum_{y_k\in\mathbf{y}} (\log Q^{\mathrm{s}}_{k,i_k,j_k} + \log Q^{\mathrm{l}}_{k,i_k,j_k,l_k}). \tag{10}$$

At test time, every segment $\hat{y}_k$ is inferred in terms of the previous predicted segments $[\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{k-1}]$. Algorithm 1 demonstrates how our proposed framework makes inference. Note that Algorithm 1 uses greedy search to get $\hat{y}$ because it is both fast in speed and effective in accuracy in our experiments, although beam search may be better in accuracy.

3 Experiments

Extensive experiments have been conducted on syntactic chunking and Chinese POS tagging across 3 datasets. Firstly, our models have obtained new state-of-the-art performances on all the datasets. Then, we have investigated ablation studies to understand the importance of each component. Lastly, case study and the results on segmenting long-length sentences confirm the effectiveness of the proposed framework in capturing the long-term dependencies among segments.

3.1 Settings

Syntactic chunking segments a word sequence into multiple labeled groups of words. We use CoNLL-2000 dataset (Sang and Buchholz, 2000), which defines 11 syntactic chunk types (NP, VP, PP, etc.). Standard data includes a training set and a test set. Following Xin et al. (2018), we randomly sample 1000 sentences from the training set as the development set. Chinese POS tagging converts a Chinese character sequence into a token sequence and associates every word with a POS tag. We use Penn Chinese Treebank 9.0 (CTB9) (Xue et al., 2005) and Universal Dependencies 1.4 (UD1) (Nivre et al., 2016). CTB9 contains the source text in various genres, covering its previous versions (e.g., CTB6). We use the Chinese section of UD1. We follow the same format and partition of the two datasets as Shao et al. (2017).

We use the same neural network configuration for all 3 datasets. The dimensions of token embedding and label embedding are respectively set as 300 and 50. The hidden unit sizes for the encoder and decoder are 256 and 512, respectively. The layers of two LSTMs are both 2. L2 regulariza-

Approach		CoNLL-2000
Segmental RNN (Kong et al., 2015)		95.08
Bi-LSTM + CRF (Huang et al., 2015)		94.46
Char-IntNet-5 (Xin et al., 2018)		95.29
GCDT (Liu et al., 2019)		95.17
Flair Embedding (Akbik et al., 2018)		96.72
Cross-view Training (Clark et al., 2018)		97.00
GCDT w/ BERT (Liu et al., 2019)		96.81
This Work	Our Model	**96.13**
	Our Model w/ BERT	**97.05**

Table 2: The performances of the baselines and our models on CoNLL-2000 dataset.

Approach		PTB9	UD1
Segmental RNN (Kong et al., 2015)		92.16	90.01
Bi-RNN + CRF (single) (Shao et al., 2017)		91.89	89.41
Bi-RNN + CRF (ensemble) (Shao et al., 2017)		92.34	89.75
Lattice LSTM (Zhang and Yang, 2018)		92.13	90.09
Glyce + Lattice LSTM (Meng et al., 2019)		92.38	90.87
BERT (Devlin et al., 2019)		92.29	94.79
Glyce + BERT (Meng et al., 2019)		93.15	96.14
This Work	Our Model	**92.56**	**91.65**
	Our Model w/ BERT	**93.38**	**96.43**

Table 3: The results on the two datasets of Chinese POS tagging.

tion is set as 1×10^{-6} and dropout ratio is set as 0.4 for reducing overfit. The above setting is obtained by grid search. We adopt Adam (Kingma and Ba, 2014) as the optimization algorithm and adopt the suggested hyper-parameters. For CoNLL-2000 dataset, the cased, 300d Glove (Pennington et al., 2014) is used to initialize token embedding. $\mathrm{CharCNN}$ is not used in Chinese tasks. The batch size is set as 16. All our models in experiments are running on NVIDIA Tesla P100.

At test time, following previous literature, we convert the prediction of our model into IOB format and use the standard conlleval script[1] to get the F1 score. We select the model that works the best on development set, and then evaluate it on test set. In all the experiments, the improvements of our models over the baselines are statistically significant with $p < 0.05$ under t-test.

3.2 Results on Syntactic Chunking

Our models are compared with two groups of baselines. One of them is trained without any external resources besides the training data:

Segmental RNN It's a segment-level model that defines a joint probability distribution over the partition of an input sequence and the labeling of the segments;

Bi-LSTM + CRF It utilizes bidirectional LSTM to read the input sentence and CRF to decode the label sequence;

Char-IntNet-5 It is a funnel-shaped CNN model with no down-sampling which learns a better internal structure for tokens;

GCDT It deepens the state transition path at each position in a sentence and assigns each token with a global representation learned from the entire sentence.

The other uses extra unlabeled corpora or fine-tunes on a pre-trained language model:

Flair Embedding It firstly pre-trains a character-level language model on a large corpus, and then uses a sequence labeling model (e.g., Bi-LSTM + CRF) to fine-tune on it;

Cross-view Training It designs a LSTM based sentence encoder to facilitate semi-supervised

[1]https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt.

Approach	CoNLL-2000	PTB9	UD1
Our Model	**96.13**	**92.56**	**91.65**
w/o CharCNN	95.81	-	-
w/o LSTM-minus, w/ Inside Algorithm	**96.35**	**92.71**	**91.87**
w/o LSTM Decoder f^d, w/ MLP	94.91	91.28	90.05
w/o Phrase Representation $\mathbf{h}^p_{i_{k-1},j_{k-1}}$ in Equation 4	95.78	92.09	91.27
w/o Label Representation $\mathbf{E}^l(l_{k-1})$ in Equation 4	95.82	91.92	91.04
w/o Greedy Search, w/ Beam Search	**96.22**	**92.77**	**91.72**

Table 4: The results of ablation experiments on all three datasets.

learning. The model can benefit from massive unlabeled corpora;

GCDT w/ BERT It adopts BERT as additional token embeddings to improve GCDT.

We adopt most of the results of baselines as reported in Huang et al. (2015); Akbik et al. (2018); Xin et al. (2018). Since the evaluation method of GCDT is not standard (see the experiment setup in Luo et al. (2020)), we correct its source code[2] to retest the performance. The result for Segmental RNN is from our re-implementation.

Table 2 demonstrates that we have notably outperformed previous methods and achieved new state-of-the-art results on CoNLL-2000 dataset. When not using external resource, we obtain the F1 score of 96.13, which outperforms Segmental RNN by 1.10%, Bi-LSTM + CRF by 1.77%, Char-IntNet-5 by 0.88%, and GCDT by 1.01%. Note that Segmental RNN, a segment-level model, underperforms Char-IntNet-5, a token-level model, by 0.22%. To make a fair comparison with the baselines using additional unlabeled corpora, we also use BERT (Devlin et al., 2019), a powerful pretrained language model, to replace our token embedding. In this way, we achieve the F1 score of 97.05, which outnumbers Flair Embedding by 0.34%, Cross-view Training by 0.05%, and GCDT w/ BERT by 0.25%. All these results verify the effectiveness of our framework.

3.3 Results on Chinese POS Tagging

We categorize the baselines into two types. The methods without using external resources:

Segmental RNN Also described in Section 3.2;

Bi-RNN + CRF It utilizes bidirectional RNN and CRF to model joint word segmentation and POS tagging. Ensemble learning is also used to improve the results;

Lattice LSTM It's a lattice-structured LSTM that encodes Chinese characters as well as all potential tokens that match a lexicon;

Glyce + Lattice LSTM It incorporates Chinese glyph information into Lattice LSTM.

Others using a pre-trained language model:

BERT It is a language model pre-training on a large corpus. It uses the representations from the last layer to predict IOB tags;

Glyce + BERT It integrates Chinese glyph information into BERT Tagging model.

We take most of the performances of baselines from Meng et al. (2019). The results for Segmental RNN are from our re-implementation.

Table 3 shows that our models have achieved state-of-the-art results on the two datasets, PTB9 and UD1. When BERT is not used, we have obtained the F1 scores of 92.56 and 91.65, which outperform Glyce + Lattice LSTM by 0.19% and 0.86% and Bi-RNN + CRF (ensemble) by 0.24% and 2.12%. Note that Segmental RNN, a segment-level model, underperforms Lattice LSTM, a token-level model, by 0.09% on UD1. When using BERT, even without incorporating Chinese glyph information, we still obtain the F1 scores of 93.38 and 96.43, which outperform Glyce + BERT by 0.25% and 0.30%. These results further confirm the effectiveness of our proposed framework.

3.4 Ablation Studies

As shown in Table 4, we conduct ablation studies to explore the impact of every component.

[2]https://github.com/Adaxry/GCDT.

Approach	1-22 (711)	23-44 (1030)	45-66 (248)	67-88 (23)	Overall
Bi-LSTM + CRF	94.01	94.76	92.98	87.09	94.23
GCDT	94.95	95.52	93.77	87.14	95.17
Our Model	**96.01**	**96.55**	**94.82**	**91.07**	**96.13**

Table 5: The F1 scores for the sentences of different length ranges.

Input Sentence		Other antibodies sparked by the preparation are of a sort rarely present in large quantities in infected or ill individuals
Output Segments	GCDT	(Other antibodies, NP) (sparked, VP) (by, PP) (the preparation, NP) (are, VP) (of, PP) (a sort, NP) **(rarely present, VP)** (in, PP) (large quantities, NP) (in, PP) (infected or ill individuals, NP)
	Our Model	(Other antibodies, NP) (sparked, VP) (by, PP) (the preparation, NP) (are, VP) (of, PP) (a sort, NP) (rarely present, ADJP) (in, PP) (large quantities, NP) (in, PP) (infected or ill individuals, NP)

Table 6: The case is from CoNLL-2000 dataset. The predicted segments of our model is correct.

Effect of Representation Learning. Following prior works (Ma and Hovy, 2016; Liu et al., 2019), we employ CharCNN to incorporate character information into word representations. By removing it, the F1 score on CoNLL-2000 decreases by 0.33%. Inside algorithm is another technique to construct phrase representations. After using it to replace LSTM-minus, the results on the three datasets are slightly improved by 0.23%, 0.16%, and 0.24%. Our implementation of inside algorithm is the same as described in Drozdov et al. (2019). Despite the slight improvements, its time complexity $\mathcal{O}(n^3)$ is too slow for both training and inference. Empirically, we find that the running time of inside algorithm is about 7 times slower than that of LSTM-minus.

Effect of Modeling the Dependencies Among Segments. LSTM decoder f^d models the long-term dependencies among segments. The prediction of every segment y_k is conditional on prior segments $y_{k'}, 1 \leq k' < k$. By replacing it with multilayer perceptron (MLP), the performances fall by 1.29%, 1.40%, and 1.78% on the three datasets. We use both phrase representation $\mathbf{h}^p_{i_{k-1},j_{k-1}}$ and label representation $\mathbf{E}^l(l_{k-1})$ to embed the previous segment y_{k-1}. After removing the phrase representations, the F1 scores decrease by 0.37%, 0.51%, and 0.42% on the three datasets. By removing the label representations, the results also drop by 0.32%, 0.70%, and 0.67%.

Effect of Inference Algorithm Beam search is a widely used technique in language generation tasks, like machine translation (Bahdanau et al., 2014; Vaswani et al., 2017; Li and Yao, 2020b) and data-to-text generation (Shen et al., 2020; Li et al., 2020c; Li and Yao, 2020a). We have attempted to use beam search (with the beam size being 5), instead of greedy search, for inference. By doing so, the F1 scores of our models increased by 0.09% on CoNLL-2000, 0.23% on PTB9, and 0.08% on UD1. While obtaining slightly better performances, the time costs for inference become intolerably high (increase about 10 times). Therefore, we adopt greedy search as the default inference algorithm.

3.5 Segmenting Long-length Sentences

Compared with token-level methods, our framework better captures the long-term dependencies among segments. Therefore, our model should be more accurate in segmenting long-length sentences. To verify this, we test the baselines and our model on the sentences of different lengths.

The study is conducted on CoNLL-2000 dataset. Bi-LSTM + CRF and GCDT are very strong baselines and have open source implementations. We use toolkit NCRFPP[3] to reproduce the performances of Bi-LSTM + CRF. We use the reproduction in Section 3.2 as the results of GCDT. Table 5 shows the experiment results. Each column name denotes the sentence length range and the case number. Our model notably outperforms prior methods in terms of long sentence length ranges. For length range 45-66, our model obtains the F1 score

[3]https://github.com/jiesutd/NCRFpp.

Approach	Time Complexity	Training Time	Evaluation Time
Segmental RNN	$\mathcal{O}(n^2\|\mathcal{L}\|^2)$	16m17s	3m01s
Bi-LSTM + CRF	$\mathcal{O}(n\|\mathcal{L}\|^2)$	3m28s	0m26s
Our Model	$\mathcal{O}(n\|\mathcal{L}\|)$	2m36s	0m20s

Table 7: Running time comparisons on CoNLL-2000 dataset.

Approach	CoNLL-2003	OntoNotes 5.0
BiLSTM-CNN-CRF (Chen et al., 2019)	91.21	87.05
GRN (Chen et al., 2019)	91.44	87.67
HCR (Luo et al., 2020)	**91.96**	**87.98**
Our Model	91.42	87.74

Table 8: The results on two NER datasets.

of 94.82, which outperforms Bi-LSTM + CRF by 1.98% and GCDT by 1.12%. For length range 67-88, our model outperforms Bi-LSTM + CRF by 4.57% and GCDT by 4.51%.

3.6 Case Study

In Table 6, we present an example extracted from the test set of CoNLL-2000. Given an input sentence, we show the prediction from a strong baseline, GCDT, and our model. The output segments from our model are consistent with the ground truth segments. The segment in bold is the one incorrectly produced by GCDT.

Understanding the structure of this sentence is very hard because the main constituents “Other antibodies", “are", and “rarely present" locate far apart. By removing the two middle phrases “sparked by the preparation" and “of a sort", the original sentence can be simplified to “Other antibodies are rarely present in large quantities in infected or ill individuals", which is very clear. Therefore, correctly predicting the segment, (rarely present, ADJP), implies that our model well captures the long-term dependencies among the segments. For GCDT, a token-level segmentation model, it mistakes “rarely present" for the verb phrase of “a sort". This may result from the following two causes:

- The adjacent segments labeled with (NP, VP) frequently appear in training data;
- the token “present" acting as a verb is much more common than as a adjective.

Both potentials indicate that token-level models can’t capture the long-term dependencies well.

3.7 Running Time Analysis

Table 7 demonstrates the running time comparison among different methods. The last two columns are respectively the running times for training (one epoch) and evaluation. We set the batch size as 16 and run all the models on 1 GPU. From the table, we can draw the following two conclusions. Firstly, Segmental RNN, a segment-level model, is very slow for both training and inference due to the high time complexity. For instance, its training and testing are respectively 6.26 and 9.05 times slower than ours. Secondly, our framework is very efficient. For example, training our model for one epoch is 1.33 times faster than training Bi-LSTM + CRF, a token-level model.

3.8 Results on NER

While our model is tailored for sequence segmentation tasks, we have also tested its performances on two widely used NER datasets, CoNLL-2003 (Sang and De Meulder, 2003) and OntoNotes 5.0 (Pradhan et al., 2013). Note that, in NER, the label correlations among adjacent segments are very weak. This seems bad for our model.

Table 8 diagrams the comparison of our model and strong NER baselines. The results of GRN and HCR are copied from Chen et al. (2019); Luo et al. (2020). For BiLSTM-CNN-CRF, its scores on CoNLL-2003 and OntoNotes 5.0 are respectively from Chen et al. (2019) and our re-implementation. From the table, we can see that our model is competitive with prior methods. For example, our model underperforms HCR by only 0.27% on OntoNotes 5.0. In particular, our F1 scores are notably higher than those of BiLSTM-CNN-CRF by 0.23% and 0.79% on the two datasets. This experiment shows that our model is applicable to general sequence labeling tasks.

4 Related Work

There are two mainstream approaches to sequence segmentation. One of them treat sequence segmentation as a sequence labeling problem by using IOB tagging scheme (Huang et al., 2015; Xin et al., 2018; Clark et al., 2018; Akbik et al., 2018; Liu et al., 2019; Li et al., 2020a). Each token in a sentence is labeled as B-tag if it's the beginning of a segment, I-tag if it is inside but not the first one within the segment, or O otherwise. This method is extensively studied by prior works and provides tons of state-of-the-art results. Xin et al. (2018) introduce a funnel-shaped convolutional architecture that learns a better internal structure for the tokens. Akbik et al. (2018) propose an efficient character-level framework that uses pretrained character embedding. Clark et al. (2018) adopt semi-supervised learning to train LSTM encoder using both labeled and unlabeled corpora. Despite the effectiveness, these models are at token level, relying on multiple token-level labels to represent a single segment. This limits their full potential to capture the long-term dependencies among segments.

The other uses transition-based systems as the backbone to incrementally segment and label an input sequence (Qian et al., 2015; Zhang et al., 2016, 2018). For example, Qian et al. (2015) design special transition actions to jointly segment, tag, and normalize a sentence. These models have many advantages, such as theoretically lower time complexity and capturing non-local features. However, they are still token-level models, which predict transition actions to shift a token from the buffer to the stack or assign a label to a span.

Recently, there is a surge of interest in developing span-based models, such as Segmental RNN (Sarawagi and Cohen, 2004; Kong et al., 2015) and LUA (Li et al., 2020b). These methods circumvent using token-level labels and directly label the phrases in a sentence. Span-based models also enjoy great popularity in language modeling (Li et al., 2020d), NER (Yu et al., 2020; Li et al., 2021), and constituent parsing (Cross and Huang, 2016; Stern et al., 2017). However, its underperforms current token-level models (see Section 3.2 and Section 3.3) and is very slow in terms of running time (see Section 3.7).

5 Conclusion

In this work, we present a novel framework to sequence segmentation that segments a sentence at segment level. The segmentation process is iterative and incremental. For every step, it determines the leftmost segment of the remaining sequence. Implementations involve LSTM-minus to extract phrase representations and RNN to model the iterations of leftmost segment determination. Extensive experiments have been conducted on syntactic chunking and Chinese POS tagging across 3 datasets. We have achieved new state-of-the-art performances on all of them. Case study and the results on segmenting long-length sentences both verify the effectiveness of our framework in modeling long-term dependencies.

Acknowledgments

This work was done when the first author did internship at Ant Group. We thank anonymous reviewers for their kind and constructive suggestions.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Hui Chen, Zijia Lin, Guiguang Ding, Jianguang Lou, Yusen Zhang, and Borje Karlsson. 2019. Grn: Gated relation network to enhance convolutional neural network for named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6236–6243.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1129–1141, Minneapolis, Minnesota. Association for Computational Linguistics.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 999–1010, New Orleans, Louisiana. Association for Computational Linguistics.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2015. Segmental recurrent neural networks. *arXiv preprint arXiv:1511.06018*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Yangming Li, lemao liu, and Shuming Shi. 2021. Empirical analysis of unlabeled entity problem in named entity recognition. In *International Conference on Learning Representations*.

Yangming Li, Han Li, Kaisheng Yao, and Xiaolong Li. 2020a. Handling rare entities for neural sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6441–6451, Online. Association for Computational Linguistics.

Yangming Li, Lemao Liu, and Shuming Shi. 2020b. Segmenting natural language sentences via lexical unit analysis. *arXiv preprint arXiv:2012.05418*.

Yangming Li and Kaisheng Yao. 2020a. Interpretable nlg for task-oriented dialogue systems with heterogeneous rendering machines. *arXiv preprint arXiv:2012.14645*.

Yangming Li and Kaisheng Yao. 2020b. Rewriter-evaluator framework for neural machine translation. *arXiv preprint arXiv:2012.05414*.

Yangming Li, Kaisheng Yao, Libo Qin, Wanxiang Che, Xiaolong Li, and Ting Liu. 2020c. Slot-consistent NLG for task-oriented dialogue systems with iterative rectification network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 97–106, Online. Association for Computational Linguistics.

Yangming Li, Kaisheng Yao, Libo Qin, Shuang Peng, Yijia Liu, and Xiaolong Li. 2020d. Span-based neural buffer: Towards efficient and effective utilization of long-distance context for neural sequence models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8277–8284.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019. GCDT: A global context enhanced deep transition architecture for sequence labeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441, Florence, Italy. Association for Computational Linguistics.

Ying Luo, Fengshun Xiao, and Hai Zhao. 2020. Hierarchical contextualized representation for named entity recognition. In *AAAI*, pages 8441–8448.

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

Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. 2019. Glyce: Glyph-vectors for chinese character representations. In *Advances in Neural Information Processing Systems*, pages 2746–2757.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Tao Qian, Yue Zhang, Meishan Zhang, Yafeng Ren, and Donghong Ji. 2015. A transition-based model for joint segmentation, pos-tagging and normalization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1837–1846.

Erik F Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. *arXiv preprint cs/0009008*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. *Advances in neural information processing systems*, 17:1185–1192.

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

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. 2020. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7155–7165, Online. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315.

Yingwei Xin, Ethan Hart, Vibhuti Mahajan, and Jean-David Ruvini. 2018. Learning better internal structure of words for sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2584–2593, Brussels, Belgium. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Meishan Zhang, Nan Yu, and Guohong Fu. 2018. A simple and effective neural model for joint word segmentation and pos tagging. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1528–1538.

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

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564, Melbourne, Australia. Association for Computational Linguistics.