# A Graph-based Model for Joint Chinese Word Segmentation and Dependency Parsing

**Hang Yan, Xipeng Qiu*, Xuanjing Huang**

School of Computer Science, Fudan University, China
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, China
{hyan19, xpqiu, xjhuang}@fudan.edu.cn

## Abstract

Chinese word segmentation and dependency parsing are two fundamental tasks for Chinese natural language processing. The dependency parsing is defined at the word-level. Therefore word segmentation is the precondition of dependency parsing, which makes dependency parsing suffer from error propagation and unable to directly make use of character-level pre-trained language models (such as BERT). In this paper, we propose a graph-based model to integrate Chinese word segmentation and dependency parsing. Different from previous transition-based joint models, our proposed model is more concise, which results in fewer efforts of feature engineering. Our graph-based joint model achieves better performance than previous joint models and state-of-the-art results in both Chinese word segmentation and dependency parsing. Additionally, when BERT is combined, our model can substantially reduce the performance gap of dependency parsing between joint models and gold-segmented word-based models. Our code is publicly available at https://github.com/fastnlp/JointCwsParser.

## 1 Introduction

Unlike English, Chinese sentences consist of continuous characters and lack obvious boundaries between Chinese words. Words are usually regarded as the minimum semantic unit, therefore Chinese word segmentation (CWS) becomes a preliminary pre-process step for downstream Chinese natural language processing (NLP). For example, the fundamental NLP task, dependency parsing, is usually defined at the word-level. To parse a Chinese sentence, the process is usually performed in the following pipeline method: word segmentation, part-of-speech (POS) tagging, and dependency parsing.

However, the pipeline method always suffers from the following limitations:

(1) Error Propagation. In the pipeline method, once some words are wrongly segmented, the subsequent POS tagging and parsing will also make mistakes. As a result, pipeline models achieve dependency scores of around $75 \sim 80\%$ (Kurita et al., 2017).

(2) Knowledge Sharing. These three tasks (word segmentation, POS tagging, and dependency parsing) are strongly related. The criterion of CWS also depends on the word's grammatical role in a sentence. Therefore, the knowledge learned from these three tasks can be shared. The knowledge of one task can help others. However, the pipeline method separately trains three models, each for a single task, and cannot fully exploit the shared knowledge among the three tasks.

A traditional solution to this error propagation problem is to use joint models (Hatori et al., 2012; Zhang et al., 2014; Kurita et al., 2017). These previous joint models mainly adopted a transition-based parsing framework to integrate the word segmentation, POS tagging, and dependency parsing. Based on standard sequential shift-reduce transitions, they design some extra actions for word segmentation and POS tagging. Although these joint models achieved better performance than the pipeline model, they still suffer from two limitations:

(1) The first is the huge search space. Compared with word-level transition parsing,

*Corresponding author.



Transactions of the Association for Computational Linguistics, vol. 8, pp. 78–92, 2020. https://doi.org/10.1162/tacl_a_00301
Action Editor: Yue Zhang. Submission batch: 9/2019; Revision batch: 11/2019; Published 3/2020.

character-level transition parsing has longer sequence of actions. The search space is huge. Therefore, it is hard to find the best transition sequence exactly. Usually, approximate strategies like greedy search or beam search are adopted in practice. However, approximate strategies do not, in general, produce an optimal solution. Although exact searching is possible within $O(n^3)$ complexity (Shi et al., 2017), due to their complexity, these models just focus on unlabeled dependency parsing, rather than labeled dependency parsing.

(2) The second is the feature engineering. These transition-based joint models rely on a detailed handcrafted feature. Although Kurita et al. (2017) introduced neural models to reduce partial efforts of feature engineering, they still require hard work on how to design and compose the word-based features from the stack and the character-based features from the buffer.

Recently, graph-based models have made significant progress for dependency parsing (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017), which fully exploit the ability of the bidirectional long short-term memory network (BiLSTM) (Hochreiter and Schmidhuber, 1997) and attention mechanism (Bahdanau et al., 2015) to capture the interactions of words in a sentence. Different from the transition-based models, the graph-based models assign a score or probability to each possible arc and then construct a maximum spanning tree from these weighted arcs.

In this paper, we propose a joint model for CWS and dependency parsing that integrates these two tasks into a unified graph-based parsing framework. Because the segmentation is a character-level task and dependency parsing is a word-level task, we first formulate these two tasks into a character-level graph-based parsing framework. In detail, our model contains (1) a deep neural network encoder, which can capture the long-term contextual features for each character—it can be a multi-layer BiLSTM or pre-trained BERT, (2) a biaffine attentional scorer (Dozat and Manning, 2017), which unifies segmentation and dependency relations at the character level. Besides, unlike the previous joint models (Hatori et al., 2012; Zhang et al., 2014; Kurita et al., 2017), our joint model does not depend on the POS tagging task.

In experiments on three popular datasets, we obtain state-of-the-art performance on CWS and dependency parsing.

In this paper, we claim four contributions:

- To the best of our knowledge, this is the first graph-based method to integrate CWS and dependency parsing both in the training phase and the decoding phase. The proposed model is very concise and easily implemented.
- Compared with the previous transition-based joint models, our proposed model is a graph-based model, which results in fewer efforts of feature engineering. Additionally, our model can deal with the labeled dependency parsing task, which is not easy for transition-based joint models.
- In experiments on datasets CTB-5, CTB-7, and CTB-9, our model achieves state-of-the-art score in joint CWS and dependency parsing, even without the POS information.
- As an added bonus, our proposed model can directly utilize the pre-trained language model BERT (Devlin et al., 2019) to boost performance significantly. The performance of many NLP tasks can be significantly enhanced when BERT was combined (Sun et al., 2019; Zhong et al., 2019). However, for Chinese, BERT is based on Chinese characters, whereas dependency parsing is conducted in the word-level. We cannot directly utilize BERT to enhance the word-level Chinese dependency parsing models. Nevertheless, by using the our proposed model, we can exploit BERT to implement CWS and dependency parsing jointly.

## 2 Related Work

To reduce the problem of error propagation and improve the low-level tasks by incorporating the knowledge from the high-level tasks, many successful joint methods have been proposed to simultaneously solve related tasks, which can be categorized into three types.

### 2.1 Joint Segmentation and POS Tagging

Because segmentation is a character-level task and POS tagging is a word-level task, an intuitive idea

is to transfer both the tasks into character-level and incorporate them in a uniform framework.

A popular method is to assign a cross-tag to each character (Ng and Low, 2004). The cross-tag is composed of a word boundary part and a POS part, for example, ''B-NN'' refers to the first character in a word with POS tag ''NN''. Thus, the joint CWS and POS tagging can be regarded as a sequence labeling problem. Following this work, Zheng et al. (2013), Chen et al. (2017), and Shao et al. (2017) utilized neural models to alleviate the efforts of feature engineering.

Another line of the joint segmentation and POS tagging method is the transition-based method (Zhang and Clark, 2008, 2010), in which the joint decoding process is regarded as a sequence of action predictions. Zhang et al. (2018) used a simple yet effective sequence-to-sequence neural model to improve the performance of the transition-based method.

### 2.2 Joint POS Tagging and Dependency Parsing

Because the POS tagging task and dependency parsing task are word-level tasks, it is more natural to combine them into a joint model.

Hatori et al. (2012) proposed a transition-based joint POS tagging and dependency parsing model and showed that the joint approach improved the accuracies of these two tasks. Yang et al. (2018) extended this model by neural models to alleviate the efforts of feature engineering.

Li et al. (2011) utilized the graph-based model to jointly optimize POS tagging and dependency parsing in a unique model. They also proposed an effective POS tag pruning method that could greatly improve the decoding efficiency.

By combining the lexicality and syntax into a unified framework, joining POS tagging and dependency parsing can improve both tagging and parsing performance over independent modeling significantly.

### 2.3 Joint Segmentation, POS Tagging, and Dependency Parsing

Compared with the above two kinds of joint tasks, it is non-trivial to incorporate all the three tasks into a joint model.

Hatori et al. (2012) first proposed a transition-based joint model for CWS, POS tagging, and dependency parsing, which stated that dependency information improved the performances of word segmentation and POS tagging. Zhang et al. (2014) expanded this work by using intra-character structures of words and found the intra-character dependencies were helpful in word segmentation and POS tagging. Zhang et al. (2015) proposed joint segmentation, POS tagging, and dependency re-ranking system. This system required a base parser to generate some candidate parsing results. Kurita et al. (2017) followed the work of Hatori et al. (2012); Zhang et al. (2014) and used the BiLSTM to extract features with *n*-gram character string embeddings as input.

A related work is the full character-level neural dependency parser (Li et al., 2018), but it focuses on character-level parsing without considering the word segmentation and word-level POS tagging and parsing. Although a heuristic method could transform the character-level parsing results to word-level, the transform strategy is tedious and the result is also worse than other joint models.

Besides, there are some joint models for constituency parsing. Qian and Liu (2012) proposed a joint inference model for word segmentation, POS tagging, and constituency parsing. However, their model did not train three tasks jointly and suffered from the decoding complexity due to the large combined search space. Wang et al. (2013) first segmented a Chinese sentence into a word lattice, and then predicted the POS tags and parsed tree based on the word lattice. A dual decomposition method was used to encourage the tagger and parser to predict agreed structures.

The above methods show that syntactic parsing can provide useful feedback to word segmentation and POS tagging and the joint inference leads to improvements in all three sub-tasks. Moreover, there is no related work on joint Chinese word segmentation and dependency parsing, without POS tagging.

## 3 Proposed Model

Previous joint methods are mainly based on the transition-based model, which modifies the standard ''shift-reduce'' operations by adding some extra operations, such as ''app'' and ''tag''. Different from previous methods, we integrate word segmentation and dependency parsing into a graph-based parsing framework, which is simpler and easily implemented.

First, we transform the word segmentation to a special arc prediction problem. For example,

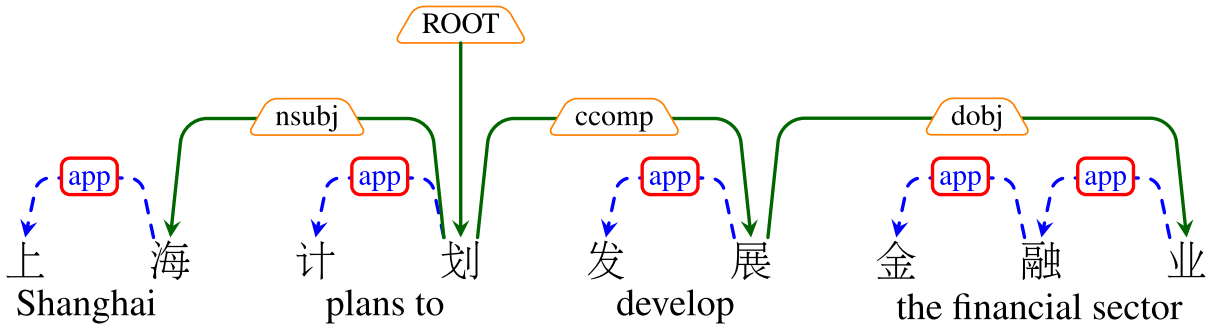


Figure 1: The unified framework for joint CWS and dependency parsing. The green arc indicates the word-level dependency relation. The dashed blue arc with ‘‘app’’ indicates its connected characters belong to a word.

a Chinese word ‘‘金融业(financial sector)’’ has two intra-word dependent arcs: ‘‘金←融’’ and ‘‘融←业’’. Both intra-word dependent arcs have the label ‘‘app’’. In this work, all characters in a word (excluding the last character) depend on their latter character, as the ‘‘金融业(financial sector)’’ in Figure 1. A character-based dependency parsing arc has also been used in Hatori et al. (2012) and Zhang et al. (2014), but their models were transition-based.

Second, we transform the word-level dependency arcs to character-level dependency arcs. Assuming that there is a dependency arc between words $w_1 = x_{i:j}$ and $w_2 = x_{u:v}$, where $x_{i:j}$ denotes the continuous characters from $i$ to $j$ in a sentence, we make this arc to connect the last characters $x_j$ and $x_v$ of each word. For example, the arc ‘‘发展(develop)→金融业(financial sector)’’ is translated to ‘‘展→业’’. Figure 1 illustrates the framework for joint CWS and dependency parsing.

Thus, we can use a graph-based parsing model to conduct these two tasks. Our model contains two main components: (1) a deep neural network encoder to extract the contextual features, which converts discrete characters into dense vectors, and (2) a biaffine attentional scorer (Dozat and Manning, 2017), which takes the hidden vectors for the given character pair as input and predicts a label score vector.

Figure 2 illustrates the model structure for joint CWS and dependency parsing. The detailed description is as follows.

### 3.1 Encoding Layer

The encoding layer is responsible for converting discrete characters into contextualized dense representations. In this paper, we tried two different kinds of encode layers. The first one is multi-layer BiLSTM, the second one is the pre-trained language model BERT (Devlin et al., 2019) which is based on self-attention.

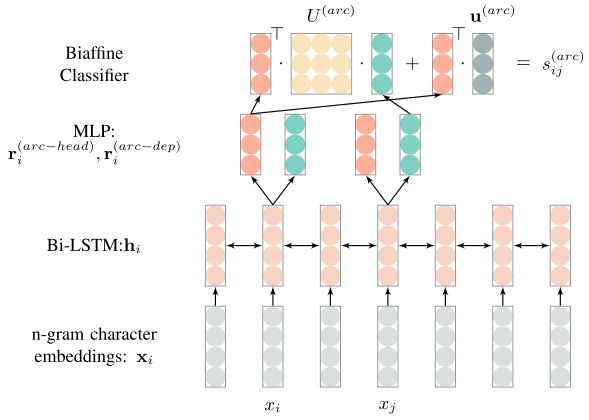


Figure 2: Proposed joint model when the encoder layer is BiLSTM. For simplicity, we omit the prediction of the arc label, which uses a different biaffine classifier.

#### 3.1.1 BiLSTM-based Encoding Layer

Given a character sequence $X = \{x_1, \ldots, x_N\}$, in neural models, the first step is to map discrete language symbols into distributed embedding space. Formally, each character $x_i$ is mapped as $\mathbf{e}_i \in \mathbb{R}^{d_e} \subset \mathbf{E}$, where $d_e$ is a hyper-parameter indicating the size of character embedding, and $\mathbf{E}$ is the embedding matrix. Character bigrams and trigrams have been shown highly effective for CWS and POS tagging in previous studies (Pei et al., 2014; Chen et al., 2015; Shao et al., 2017; Zhang et al., 2018). Following their settings, we combine the character bigram and trigram to enhance the representation of each character. The

final character representation of $x_i$ is given by $\mathbf{e}_i = \mathbf{e}_{x_i} \oplus \mathbf{e}_{x_i x_{i+1}} \oplus \mathbf{e}_{x_i x_{i+1} x_{i+2}}$, where $\mathbf{e}$ denotes the embedding for unigram, bigram, and trigram, and $\oplus$ is the concatenation operator.

To capture the long-term contextual information, we use a deep BiLSTM (Hochreiter and Schmidhuber, 1997) to incorporate information from both sides of a sequence, which is a prevalent choice in recent research for NLP tasks.

The hidden state of LSTM for the $i$-th character is

$$\mathbf{h}_i = \text{BiLSTM}(\mathbf{e}_i, \overrightarrow{\mathbf{h}}_{i-1}, \overleftarrow{\mathbf{h}}_{i+1}, \theta), \quad (1)$$

where $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are the hidden states at position $i$ of the forward and backward LSTMs respectively, and $\theta$ denotes all the parameters in the BiLSTM layer.

#### 3.1.2 BERT-based Encoding Layer

Other than using BiLSTM as the encoder layer, pre-trained BERT can also been used as the encoding layer (Devlin et al., 2019; Cui et al., 2019). The input of BERT is the character sequence $X = \{x_1, \ldots, x_N\}$, the output of the last layer of BERT is used as the representation of characters. More details on the structure of BERT can be found in Devlin et al. (2019).

### 3.2 Biaffine Layer

To predict the relations of each character pair, we use the biaffine attention mechanism (Dozat and Manning, 2017) to score their probability on the top of encoding layers. According to Dozat and Manning (2017), biaffine attention is more effectively capable of measuring the relationship between two elementary units.

#### 3.2.1 Unlabeled Arc Prediction

For the pair of the $i$-th and $j$-th characters, we first take the output of the encoding layer $\mathbf{h}_i$ and $\mathbf{h}_j$, then feed them into an extension of bilinear transformation called a *biaffine function* to obtain the score for an arc from $x_i$ (head) to $x_j$ (dependent).

$$\mathbf{r}_i^{(arc-head)} = \text{MLP}^{(arc-head)}(\mathbf{h}_i), \quad (2)$$

$$\mathbf{r}_j^{(arc-dep)} = \text{MLP}^{(arc-dep)}(\mathbf{h}_j), \quad (3)$$

$$s_{ij}^{(arc)} = \mathbf{r}_i^{(arc-head)} U^{(arc)} \mathbf{r}_j^{(arc-dep)} + \mathbf{r}_i^{(arc-head)^{\mathrm{T}}} \mathbf{u}^{(arc)}, \quad (4)$$

where MLP is a multi-layer perceptron. A weight matrix $U^{(arc)}$ determines the strength of a link from $x_i$ to $x_j$ while $\mathbf{u}^{(arc)}$ is used in the bias term, which controls the prior headedness of $x_i$.

Thus, $\mathbf{s}_j^{(arc)} = [s_{1j}^{(arc)}; \cdots; s_{Tj}^{(arc)}]$ is the scores of the potential heads of the $j$-th character, then a softmax function is applied to obtain the probability distribution.

In the training phase, we minimize the cross-entropy of golden head-dependent pair. In the test phase, we ensure that the resulting parse is a well-formed tree by the heuristics formulated in Dozat and Manning (2017).

#### 3.2.2 Arc Label Prediction

After obtaining the best predicted unlabeled tree, we assign the label scores $\mathbf{s}_{ij}^{(label)} \in \mathbb{R}^K$ for every arc $x_i \rightarrow x_j$, in which the $k$-th element corresponds to the score of $k$-th label and $K$ is the size of the label set. In our joint model, the arc label set consists of the standard word-level dependency labels and a special label ‘‘app’’ indicating the intra-dependency within a word.

For the arc $x_i \rightarrow x_j$, we obtain $\mathbf{s}_{ij}^{(label)}$ with

$$\mathbf{r}_i^{(label-head)} = \text{MLP}^{(label-head)}(\mathbf{h}_i), \quad (5)$$

$$\mathbf{r}_j^{(label-dep)} = \text{MLP}^{(label-dep)}(\mathbf{h}_j), \quad (6)$$

$$\mathbf{r}_{ij}^{(label)} = \mathbf{r}_i^{(label-head)} \oplus \mathbf{r}_j^{(label-dep)}, \quad (7)$$

$$\mathbf{s}_{ij}^{(label)} = \mathbf{r}_i^{(label-head)} \mathcal{U}^{(label)} \mathbf{r}_j^{(label-dep)} + W^{(label)}(\mathbf{r}_{ij}^{(label)}) + \mathbf{u}^{(label)}, \quad (8)$$

where $\mathcal{U}^{(label)} \in \mathbb{R}^{K \times p \times p}$ is a third-order tensor, $W^{(label)} \in \mathbb{R}^{K \times 2p}$ is a weight matrix, and $\mathbf{u}^{(label)} \in \mathbb{R}^K$ is a bias vector. The best label of arc $x_i \rightarrow x_j$ is determined according to $\mathbf{s}_{ij}^{(label)}$.

$$y_{ij} = \arg\max_{label} \mathbf{s}_{ij}^{(label)} \quad (9)$$

In the training phase, we use golden head-dependent relations and cross-entropy to optimize arc label prediction. Characters with continuous ‘‘app’’ arcs can be combined into a single word. If a character has no leftward ‘‘app’’ arc, it is a single-character word. The arc with label ‘‘app’’ is constrained to occur in two adjacent characters and is leftward. When decoding, we first use the proposed model to predict the character-level labeled dependency tree, and then recover the word segmentation and word-level dependency

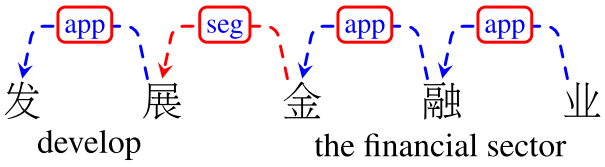


Figure 3: Label prediction for word segmentation only. The arc with ‘‘app’’ indicates its connected characters belong to a word, and the arc with ‘‘seg’’ indicates its connected characters belong to different words.

tree based on the predicted character-level arc labels. The characters with continuous ‘‘app’’ are regarded as one word. And the predicted head character of the last character is viewed as this word's head. Because the predicted arc points to a character, we regard the word that contains this head character as the head word.

### 3.3 Models for Word Segmentation Only

The proposed model can be also used for the CWS task solely. Without considering the parsing task, we first assign a leftward unlabeled arc by default for every two adjacent characters, and then predict the arc labels that indicate the boundary of segmentation. In the task of word segmentation only, there are two kinds of arc labels: ‘‘seg’’ and ‘‘app’’. ‘‘seg’’ means there is a segmentation between its connected characters, and ‘‘app’’ means its connected characters belong to one word. Because the unlabeled arcs are assigned in advance, we just use Eq. (5) $\sim$ (8) to predict the labels: ‘‘seg’’ and ‘‘app’’. Thus, the word segmentation task is transformed into a binary classification problem.

Figure 3 gives an illustration of the labeled arcs for the task of word segmentation only.

## 4 Experiments

### 4.1 Datasets

We use the Penn Chinese Treebank 5.0 (CTB-5),[1] 7.0 (CTB-7),[2] and 9.0 (CTB-9)[3] datasets to evaluate our models (Xue et al., 2005). For CTB-5, the training set is from sections 1$\sim$270, 400$\sim$931, and 1001$\sim$1151, the development set is from section 301$\sim$325, and the test set is from section 271$\sim$300; this splitting was also adopted by Zhang and Clark (2010), Zhang et al. (2014), and Kurita et al. (2017). For CTB-7, we use the same split as Wang et al. (2011), Zhang et al. (2014), and Kurita et al. (2017). For CTB-9, we use the dev and test files proposed by Shao et al. (2017), and we regard all left files as the training data.

[1] https://catalog.ldc.upenn.edu/LDC2005T01.
[2] https://catalog.ldc.upenn.edu/LDC2010T07.
[3] https://catalog.ldc.upenn.edu/LDC2016T13.

### 4.2 Measures

Following Hatori et al. (2012), Zhang et al. (2014, 2015), and Kurita et al. (2017), we use standard measures of word-level F1, precision, and recall scores to evaluate word segmentation and dependency parsing (for both unlabeled and labeled scenario) tasks. We detail them in the following.

- $F1_{seg}$: F1 measure of CWS. This is the standard metric used in the CWS task (Qiu et al., 2013; Chen et al., 2017).

- $F1_{udep}$: F1 measure of unlabeled dependency parsing. Following Hatori et al. (2012), Zhang et al. (2014, 2015), and Kurita et al. (2017), we use standard measures of word-level F1, precision, and recall score to evaluate dependency parsing. In the scenario of joint word segmentation and dependency parsing, the widely used unlabeled attachment score (UAS) is not enough to measure the performance, since the error arises from two aspects: One is caused by word segmentation and the other is due to the wrong prediction on the head word. A dependent-head pair is correct only when both the dependent and head words are accurately segmented and the dependent word correctly finds its head word. The precision of unlabeled dependency parsing (denoted as $P_{udep}$) is calculated by the correct dependent-head pair versus the total number of dependent-head pairs (namely the number of segmented words). The recall of unlabeled dependency parsing (denoted as $R_{udep}$) is computed by the correct dependent-head pair versus the total number of golden dependent-head pairs (namely, the number of golden words). The calculation of $F1_{udep}$ is like $F1_{seg}$.

- $F1_{ldep}$: F1 measure of labeled dependency parsing. The only difference from $F1_{udep}$ is that except for the match between the head and dependent words, the pair must have the same label as the golden dependent-head pair. The precision and recall are calculated correspondingly. Because the number of

golden labeled dependent-head pairs and predicted labeled dependent-head pairs are the same with the counterparts of unlabeled dependency parsing, the value of $F1_{ldep}$ cannot be higher than $F1_{udep}$.

A more detailed description of dependency parsing metrics can be found in Kübler et al. (2009). The $UAS$, $LAS$ equal to the value of the recall of unlabeled dependency parsing ($R_{udep}$) and the recall of labeled dependency parsing ($R_{ldep}$), respectively. We also report these two values in our experiments.

### 4.3 Experimental Settings

**Pre-trained Embedding** Based on Shao et al. (2017); Zhang et al. (2018), *n*-grams are of great benefit to CWS and POS tagging tasks. Thus we use unigram, bigram, and trigram embeddings for all of our character-based models. We first pre-train unigram, bigram, and trigram embeddings on the Chinese Wikipedia corpus by the method proposed in Ling et al. (2015), which improves standard Word2Vec by incorporating token order information. For a sentence with characters ''abcd...'', the unigram sequence is ''a b c ...''; the bigram sequence is ''ab bc cd ...''; and the trigram sequence is ''abc bcd ...''. For our word dependency parser, we use Tencent's pre-trained word embeddings (Song et al., 2018). Because Tencent's pre-trained word embedding dimension is 200, we set both pre-trained and random word embedding dimension as 200 for all of our word dependency parsing models. All pre-trained embeddings are fixed during our experiments. In addition to the fixed pre-trained embeddings, we also randomly initialize embeddings, and element-wisely add the pre-trained and random embeddings before other procedures. For a model with BERT encoding layer, we use the Chinese BERT-base released in Cui et al. (2019).

**Hyper-parameters** The development set is used for parameter tuning. All random weights are initialized by Xavier normal initializer (Glorot and Bengio, 2010).

For BiLSTM based models, we generally follow the hyper-parameters chosen in Dozat and Manning (2017). The model is trained with the Adam algorithm (Kingma and Ba, 2015) to minimize the sum of the cross-entropy of arc predictions and label predictions. After each training epoch, we test the model on the dev set, and models with the highest $F1_{udep}$ in development set are used to evaluate on the test sets; the results reported for different datasets in this paper are all on their test set. Detailed hyper-parameters can be found in Table 1.

| | |
|---|---|
| Embedding dimension | 100 |
| BiLSTM hidden size | 400 |
| Gradients clip | 5 |
| Batch size | 128 |
| Embedding dropout | 0.33 |
| LSTM dropout | 0.33 |
| Arc MLP dropout | 0.33 |
| Label MLP dropout | 0.33 |
| LSTM depth | 3 |
| MLP depth | 1 |
| Arc MLP size | 500 |
| Label MLP size | 100 |
| Learning rate | 2e-3 |
| Annealing | $.75^{t/5000}$ |
| $\beta_1, \beta_2$ | 0.9 |
| Max epochs | 100 |

Table 1: Hyper-parameter settings.

For BERT based models, we use the AdamW optimizer with a triangle learning rate warmup, the maximum learning rate is $2e-5$ (Loshchilov and Hutter, 2019; Devlin et al., 2019). It optimizes for five epochs, the model with the best development set performance is used to evaluate on the test sets.

### 4.4 Proposed Models

In this section, we introduce the settings for our proposed joint models. Based on the way the model uses dependency parsing labels and encoding layers, we divide our models into four kinds. We enumerate them as follows.

- **Joint-SegOnly model**: The proposed model can be also used for word segmentation task only. In this scenario, the dependency arcs are just allowed to appear in two adjacent characters and $label \in \{app, seg\}$. This model is described in Section 3.3.

- **Joint-Binary model**: This scenario means $label \in \{app, dep\}$. In this situation, the label information of all the dependency arcs is ignored. Each word-level dependency arc is labeled as $dep$, the intra-word dependency is regarded as $app$. Characters with

| Models | CTB-5 | | CTB-7 | | CTB-9 | |
|---|---|---|---|---|---|---|
| | $F1_{seg}$ | $F1_{udep}$ | $F1_{seg}$ | $F1_{udep}$ | $F1_{seg}$ | $F1_{udep}$ |
| Hatori et al. (2012) | 97.75 | 81.56 | 95.42 | 73.58 | – | – |
| Zhang et al. (2014) STD | 97.67 | 81.63 | 95.53 | 75.63 | – | – |
| Zhang et al. (2014) EAG | 97.76 | 81.70 | 95.39 | 75.56 | – | – |
| Zhang et al. (2015) | 98.04 | 82.01 | – | – | – | – |
| Kurita et al. (2017) | 98.37 | 81.42 | 95.86 | 74.04 | – | – |
| Joint-Binary | 98.45 | 87.24 | 96.57 | 81.34 | 97.10 | 81.67 |
| Joint-Multi | **98.48** | **87.86** | **96.64** | **81.80** | **97.20** | **82.15** |
| Joint-Multi-BERT | 98.46 | **89.59** | **97.06** | **85.06** | **97.63** | **85.66** |

STD and EAG in Zhang et al. (2014) denote the arc-standard and the arc-eager models.
$F1_{seg}$ and $F1_{udep}$ are the F1 score for Chinese word segmentation and unlabeled dependency parsing, respectively.

Table 2: Main results in the test set of different datasets. Our Joint-Multi model achieves superior performance than previous joint models. The Joint-Multi-BERT further enhances the performance of dependency parsing significantly.



continuous $app$ label will be joined together and viewed as one word. The $dep$ label indicates this character is the end of a word.

- **Joint-Multi model**: This scenario means $label \in \{app, dep_1, \cdots, dep_K\}$, where $K$ is the number of types of dependency arcs. The intra-word dependency is viewed as $app$. The other labels are the same as the original arc labels. But instead of representing the relationship between two words, the labeled arc represents the relationship between the last character of the dependent word and the last character of the head word.

- **Joint-Multi-BERT model**: For this kind of model, the encoding layer is BERT. And it uses the same target scenario as the Joint-Multi model.

### 4.5 Comparison with the Previous Joint Models

In this section, we mainly focus on the performance comparison between our proposed models and the previous joint models. Because the previous models just deal with the unlabeled dependency parsing, we just report the $F1_{seg}$ and $F1_{udep}$ here.

As presented in Table 2, our model (**Joint-Binary**) outpaces previous methods with a large margin in both CWS and dependency parsing, even without the local parsing features that were extensively used in previous transition-based joint work (Hatori et al., 2012; Zhang et al., 2014, 2015; Kurita et al., 2017). Another difference between our joint models and previous works is the combination of POS tags; the previous models all used the POS task as one componential task. Despite the lack of POS tag information, our models still achieve much better results. However, according to Dozat and Manning (2017), POS tags are beneficial to dependency parsers, therefore one promising direction of our joint model might be incorporating POS tasks into this joint model.

Other than the performance distinction between previous work, our joint model with or without dependency labels also differ from each other. It is clearly shown in Table 2 that our joint model with labeled dependency parsing (**Joint-Multi**) outperforms its counterpart (**Joint-Binary**) in both CWS and dependency parsing. With respect to the enhancement of dependency parsing caused by the arc labels, we believe it can be credited to two aspects. The first one is the more accurate CWS. The second one is that label information between two characters will give extra supervision for the search of head characters. The reason why labeled dependency parsing is conducive to CWS will be also analyzed in Section 4.6.

Owing to the joint decoding of CWS and dependency parsing, we can utilize the character-level pre-trained language model BERT. The last row of Table 2 displays that the $F1_{udep}$ can be substantially increased when BERT is used, even when the performance of CWS not improve too much. We presume this indicates that BERT can better extract the contextualized information to help the dependency parsing.

| Models | Tag Set | CTB-5 | | | CTB-7 | | | CTB-9 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $F1_{seg}$ | $P_{seg}$ | $R_{seg}$ | $F1_{seg}$ | $P_{seg}$ | $R_{seg}$ | $F1_{seg}$ | $P_{seg}$ | $R_{seg}$ |
| LSTM+MLP | $\{B, M, E, S\}$ | 98.47 | 98.26 | 98.69 | 95.45 | 96.44 | 96.45 | 97.11 | 97.19 | 97.04 |
| LSTM+CRF | $\{B, M, E, S\}$ | 98.48 | 98.33 | 98.63 | 96.46 | 96.45 | 96.47 | 97.15 | 97.18 | 97.12 |
| LSTM+MLP | $\{app, seg\}$ | 98.40 | 98.14 | 98.66 | 96.41 | 96.53 | 96.29 | 97.09 | 97.16 | 97.02 |
| Joint-SegOnly | $\{app, seg\}$ | **98.50** | **98.30** | 98.71 | 96.50 | 96.67 | 96.34 | 97.09 | 97.15 | 97.04 |
| Joint-Binary | $\{app, dep\}$ | 98.45 | 98.16 | 98.74 | 96.57 | 96.66 | 96.49 | 97.10 | 97.16 | 97.04 |
| Joint-Multi | $\{app, dep_1, \cdots, dep_K\}$ | 98.48 | 98.17 | **98.80** | **96.64** | **96.68** | **96.60** | **97.20** | **97.31** | **97.19** |
| Joint-Multi-BERT | $\{app, dep_1, \cdots, dep_K\}$ | 98.46 | 98.12 | **98.81** | **97.06** | **97.05** | **97.08** | **97.63** | **97.68** | **97.58** |

The upper part refers the models based on sequence labeling.
The lower part refers our proposed joint models which are detailed in Section 4.4. The proposed joint models achieve near or better $F1_{seg}$ than models trained only on Chinese word segmentation.
$F1_{seg}$, $P_{seg}$, and $R_{seg}$ are the F1, precision, and recall of CWS, respectively.

Table 3: Results of Chinese word segmentation.

### 4.6 Chinese Word Segmentation

In this part, we focus on the performance of our model for the CWS task only.

Most of state-of-the-art CWS methods are based on sequence labeling, in which every sentence is transformed into a sequence of $\{B, M, E, S\}$ tags. $B$ represents the begin of a word, $M$ represents the middle of a word, $E$ represents the end of a word, and $S$ represents a word with only one character. We compare our model with these state-of-the-art methods.

- LSTM+MLP with $\{B, M, E, S\}$ tags. Following Ma et al. (2018), we tried to do CWS without conditional random fields (CRFs). After BiLSTM, the hidden states of each character further forwards into a multi-layer perceptron (MLP), so that every character can output a probability distribution over the label set. The Viterbi algorithm is utilized to find the global maximum label sequence when testing.

- LSTM+CRF with $\{B, M, E, S\}$ tags. The only difference between this scenario and the previous one is whether using CRF after the MLP (Lafferty et al., 2001; Chen et al., 2017).

- LSTM+MLP with $\{app, seg\}$ tags. The segmentation of a Chinese sentence can be represented by a sequence of $\{app, seg\}$, where $app$ represents that the next character and this character belongs to the same word, and $seg$ represents that this character is the last character of a word. Therefore, CWS can be viewed as a binary classification problem. Except for the tag set, this model's architecture is similar to the LSTM+MLP scenario.

All of these models use the multi-layer BiLSTM as the encoder; they differ from each other in their way of decoding and the tag set. The number of BiLSTM layers is 3 and the hidden size is 200.

The performance of all models are listed in Table 3. The first two rows present the difference between whether utilizing CRF on the top of MLP. CRF performance is slightly better than its counterpart. The first row and the third row display the comparison between different tag scenarios, the $\{B, M, E, S\}$ tag set is slightly better than the $\{app, seg\}$ tag set.

Different from the competitor sequence labeling model (LSTM+MLP with $\{app, seg\}$ tag set), our joint-SegOnly model uses the biaffine to model the interaction between the two adjacent characters near the boundary and achieves slightly better or similar performances on all datasets. The empirical results in the three datasets suggest that modeling the interaction between two consecutive characters are helpful to CWS. If two characters are of high probability to be in a certain dependency parsing relationship, there will be a greater chance that one of the characters is the head character.

The lower part of Table 3 shows the segmentation evaluation of the proposed joint models. Jointly training CWS and dependency parsing achieves comparable or slightly better CWS than training CWS alone. Although head prediction is not directly related to CWS, the head character can

| Models | CTB-5 | | | | | CTB-7 | | | | | CTB-9 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F1_{seg}$ | $F1_{udep}$ | $UAS$ | $F1_{ldep}$ | $LAS$ | $F1_{seg}$ | $F1_{udep}$ | $UAS$ | $F1_{ldep}$ | $LAS$ | $F1_{seg}$ | $F1_{udep}$ | $UAS$ | $F1_{ldep}$ | $LAS$ |
| Biaffine[†] | – | – | 88.81 | – | 85.63 | – | – | 86.06 | – | 81.33 | – | – | 86.21 | – | 81.57 |
| Pipeline[§] | **98.50** | 86.50 | 86.71 | 83.46 | 83.67 | 96.50 | 80.62 | 80.49 | 76.58 | 76.46 | 97.09 | 81.54 | 81.61 | 77.34 | 77.40 |
| Joint-Multi | 98.48 | **87.86** | **88.08** | **85.08** | **85.23** | **96.64** | **81.80** | **81.80** | **77.84** | **77.83** | **97.20** | **82.15** | **82.23** | **78.08** | **78.14** |
| Joint-Multi-BERT | 98.46 | **89.59** | **89.97** | **85.94** | **86.3** | **97.06** | **85.06** | **85.12** | **80.71** | **80.76** | **97.63** | **85.66** | **85.74** | **81.71** | **81.77** |

[†] The results are evaluated by a word-level biaffine parser on the gold-segmented sentences.
[§] The pipeline model first uses the Joint-SegOnly model to segment the sentence, then uses the word-level biaffine parser to obtain the parsing result.

Table 4: Comparison with the pipeline model. Our Joint-Multi models outperform the pipeline models in a large margin. When BERT is used, the dependency parsing performance was significantly improved, although the Chinese word segmentation does not meliorate a lot.

| Models | CTB-5 | | | | | CTB-7 | | | | | CTB-9 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F1_{seg}$ | $F1_{udep}$ | $UAS$ | $F1_{ldep}$ | $LAS$ | $F1_{seg}$ | $F1_{udep}$ | $UAS$ | $F1_{ldep}$ | $LAS$ | $F1_{seg}$ | $F1_{udep}$ | $UAS$ | $F1_{ldep}$ | $LAS$ |
| Joint-Multi | 98.48 | 87.86 | 88.08 | 85.08 | 85.23 | 96.64 | 81.80 | 81.80 | 77.84 | 77.83 | 97.20 | 82.15 | 82.23 | 78.08 | 78.14 |
| -pre-trained | 97.72 | 82.56 | 82.70 | 79.8 | 70.93 | 95.52 | 76.35 | 76.22 | 72.16 | 72.04 | 96.56 | 78.93 | 78.93 | 74.35 | 74.37 |
| -*n*-gram | 97.72 | 83.44 | 83.60 | 80.24 | 80.41 | 95.21 | 77.37 | 77.11 | 72.94 | 72.69 | 95.85 | 78.55 | 78.41 | 73.94 | 73.81 |

The '-pre-trained' means the model is trained without the pre-trained embeddings.
The '-*n*-gram' means the model is trained by removing the bigram and trigram embeddings, only randomly initialized and pre-trained character embeddings are used.

Table 5: Ablation experiments for Joint-Multi models.

only be the end of a word, therefore combination between CWS and character dependency parsing actually introduces more supervision for the former task. On CTB-5, the Joint-Binary and Joint-Multi models are slightly worse than the Joint-SegOnly model. The reason may be that the CTB-5 dataset is relatively small and the complicated models suffer from the overfitting. From the last row of Table 3, BERT can further enhance the model's performance on CWS.

Another noticeable phenomenon from the lower part of Table 3 is that the labeled dependency parsing brings benefit to CWS. We assume this is because the extra supervision from dependency parsing labels is informative for word segmentation.

### 4.7 Comparison with the Pipeline Model

In this part, we compare our joint model with the pipeline model. The pipeline model first uses our best Joint-SegOnly model to obtain segmentation results, then applies the word-based biaffine parser to parse the segmented sentence. The word-level biaffine parser is the same as in Dozat and Manning (2017) but without POS tags. Just like the joint parsing metric, for a dependent-head word pair, only when both head and dependent words are correct can this pair be viewed as a right one.

Table 4 obviously shows that in CTB-5, CTB-7, and CTB-9, the Joint-Multi model consistently outperforms the pipeline model in $F1_{udep}$, $UAS$, $F1_{ldep}$, and $LAS$. Although the $F1_{seg}$ difference between the Joint-Multi model and the pipeline model is only $-0.02, +0.14, +0.11$ in CTB-5, CTB-7, and CTB-9, respectively, the $F1_{udep}$ of the Joint-Multi is higher than the pipeline model by $+1.36, +1.18$, and $+0.61$, respectively; we believe this indicates the better resistance to error propagation of the Joint-Multi model.

Additionally, when BERT is used, $F1_{udep}$, $UAS$, $F1_{ldep}$, and $LAS$ are substantially improved, which represents that our joint model can take advantage of the power of BERT. In CTB-5, the joint model even achieves better $UAS$ than the gold-segmented word-based model. And for the $LAS$, Joint-Multi-BERT models also achieve better results in CTB-5 and CTB-9. We presume the reason that performance of CWS does not improve as much as dependency parsing is that the falsely segmented words in Joint-Multi-BERT are mainly segmenting a long word into several short words or recognizing several short words as one long word.

### 4.8 Ablation Study

As our model uses various *n*-gram pre-trained embeddings, we explore the influence of these pre-trained embeddings. The second row in Table 5 shows the results of the Joint-Multi model without pre-trained embeddings; it is clear that pre-trained

| Models | CTB-5 | | | CTB-7 | | | CTB-9 | | |
|---|---|---|---|---|---|---|---|---|---|
| | $P_{udep}$ | Seg-wrong | Head-wrong | $P_{udep}$ | Seg-wrong | Head-wrong | $P_{udep}$ | Seg-wrong | Head-wrong |
| Pipeline | 86.28% | 3.49% | 10.23% | 80.75% | 7.10% | 12.15% | 81.48% | 6.76% | 11.76% |
| Joint-Multi | 87.65% | 3.43% | 8.92% | 81.81% | 6.80% | 11.39% | 82.08% | 6.55% | 11.37% |
| Joint-Multi-BERT | 89.22% | 3.2% | 7.58% | 85.01% | 6.10% | 8.89% | 85.59% | 5.74% | 8.89% |

The value of $P_{udep}$ is the percentage that the predicted arc is correct. 'Seg-wrong' means that either head or dependent (or both) is wrongly segmented. 'Head-wrong' means that the word is correctly segmented but the predicted head word is wrong.

Table 6: Error analysis of unlabeled dependency parsing in the test set of different datasets.

embeddings are important for both the word segmentation and dependency parsing.

We also tried to remove the bigram and trigram. Results are illustrated in the third row of Table 5. Compared with the Joint-Multi model, without bigram and trigram, it performs worse in all metrics. However, the comparison between the second row and the third row shows divergence in CWS and dependency parsing for datasets CTB-5 and CTB-7. For CWS, the model without pre-trained embeddings obtains superior performance than without the bigram and trigram feature, whereas for all dependency parsing related metrics, the model with pre-trained character embedding obtains better performance. We assume the *n*-gram features are important to Chinese word segmentation. For the dependency parsing task, however, the relation between two characters are more beneficial; when pre-trained embeddings are combined, the model can exploit the relationship encoded in the pre-trained embeddings. Additionally, for CTB-5 and CTB-7, even though the third row has inferior $F1_{seg}$ (on average $0.16\%$ lower than the second row), it still achieves much better $F1_{udep}$ (on average $0.95\%$ higher than the second row). We believe this is a proof that joint CWS and dependency parsing is resistant to error propagation. The higher segmentation and dependency parsing performance for the model without pre-trained embedding in CTB-9 might be owing to its large training set, which can achieve better results even from randomly initialized embeddings.

### 4.9 Error Analysis

Apart from performing the standard evaluation, we investigate where the dependency parsing head prediction error comes from. The errors can be divided into two kinds, (1) either the head or dependent (or both) is wrongly segmented, or (2) there is the wrong choice on the head word. The ratio of these two mistakes is presented in Table 6. For the Joint-Multi model, more mistakes caused by segmentation in CTB-7 is coherent with our observation that CTB-7 bears lower CWS performance. Based on our error analysis, the wrong prediction of head word accounts for most of the errors, therefore further joint models addressing head prediction error problem might result in more gain on performance.

Additionally, although from Table 4 the distinction of $F1_{seg}$ between the Joint-Multi model and the Pipeline model is around $+0.1\%$ on average, the difference between the Head-wrong is more than around $+0.82\%$ in average. We think this is caused by the pipeline model, in which is more sensitive to word segmentation errors and suffers more from the OOV problem, as depicted in Figure 4. From the last row of Table 4, Joint-Multi-BERT achieves excellent performance on dependency parsing because it can significantly reduce errors caused by predicting the wrong head.

## 5 Conclusion and Future Work

In this paper, we propose a graph-based model for joint Chinese word segmentation and dependency parsing. Different from the previous joint models, our proposed model is a graph-based model and is more concise, which results in fewer efforts of feature engineering. Although no explicit handcrafted parsing features are applied, our joint model outperforms the previous feature-riched joint models by a large margin. The empirical results in CTB-5, CTB-7, and CTB-9 show that the dependency parsing task is also beneficial to Chinese word segmentation. Additionally, labeled dependency parsing not only is good for Chinese word segmentation, but also avails the dependency parsing head prediction.

Apart from good performance, the comparison between our joint model and the pipeline model

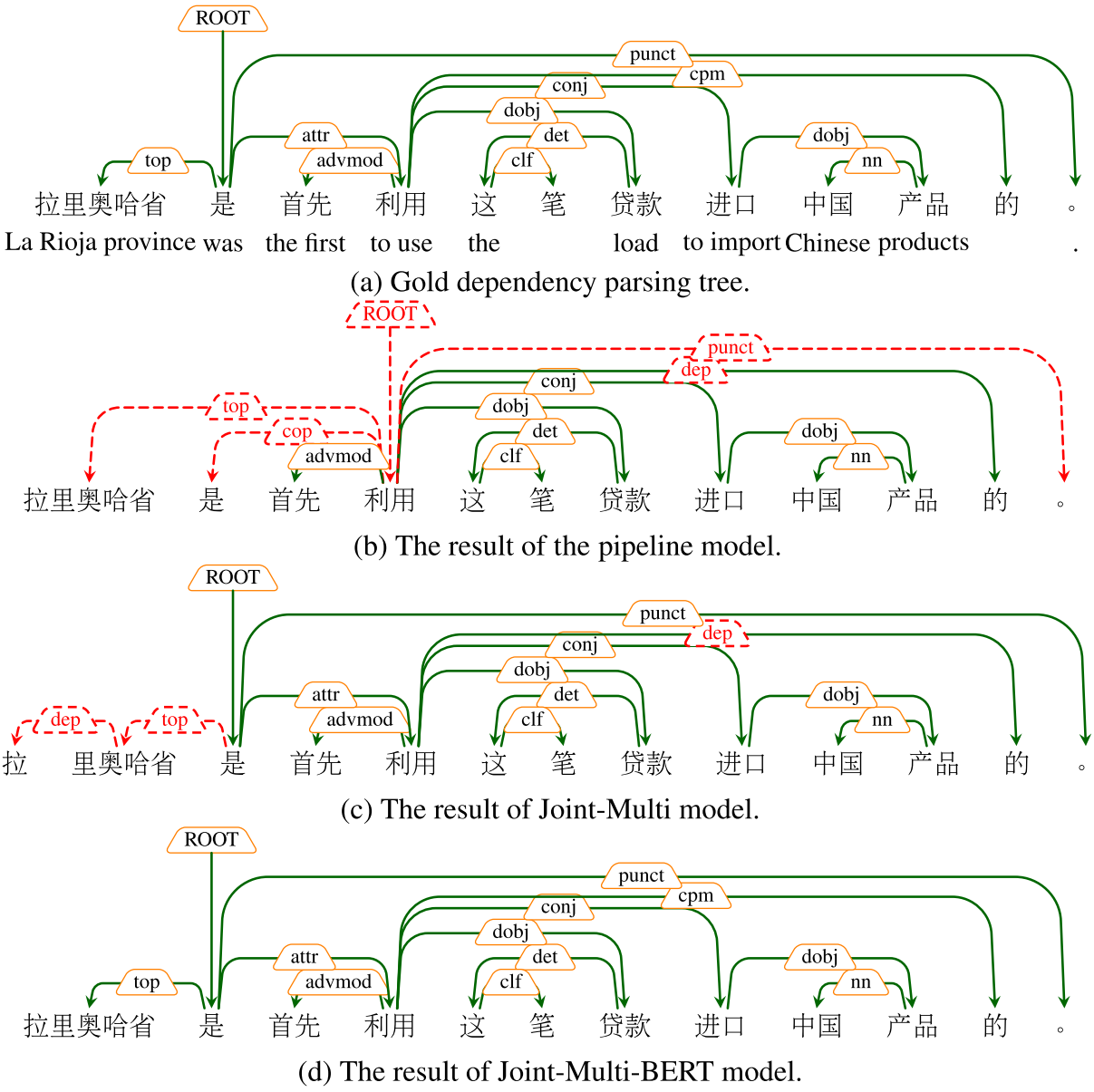


Figure 4: Parsing results of different models. The red dashed box means the dependency label is wrong. The red dashed edge means this dependency arc does not exist. Although the pipeline model has the right word segmentation, ‘‘拉里奥哈省’’ is an out-of-vocabulary word. Therefore, it fails to find the right dependency relation and adversely affects predictions afterward. The Joint-Multi model can still have a satisfying outcome even with wrong segmentation, which depicts that the Joint-Multi model is resistant to wrong word segmentations. The Joint-Multi-BERT correctly finds the word segmentation and dependency parsing.

shows great potentialities for character-based Chinese dependency parsing. And owing to the joint decoding between Chinese word segmentation and dependency parsing, our model can use a pre-trained character-level language model (such as BERT) to enhance the performance further. After the incorporation of BERT, the performance of our joint model increases substantially, resulting in the character-based dependency parsing performing near the gold-segmented word-based dependency parsing. Our proposed method not merely outpaces the pipeline model, but also avoids the preparation for pre-trained word embeddings that depends on a good Chinese word segmentation model.

In order to fully explore the possibility of graph-based Chinese dependency parsing, future work should be done to incorporate POS tagging into this framework. Additionally, as illustrated in Zhang et al. (2014), a more reasonable intra-word dependent structure might further boost the performance of all tasks.

## Acknowledgments

We would like to thank the action editor and the anonymous reviewers for their insightful comments. We also thank the developers of fastNLP,[4] Yunfan Shao and Yining Zheng, for developing this handy natural language processing package. This work was supported by the National Key Research and Development Program of China (no. 2018YFC0831103), National Natural Science Foundation of China (no. 61672162), Shanghai Municipal Science and Technology Major Project (no. 2018SHZDZX01), and ZJLab.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2017. A feature-enriched neural model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19–25, 2017*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long Short-Term Memory Neural Networks for Chinese Word Segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17–21, 2015*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for Chinese BERT. *CoRR*, abs/1906.08101v2.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13–15, 2010*.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8–14, 2012, Jeju Island, Korea - Volume 1: Long Papers*.

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics TACL*, 4:313–327.

Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Shuhei Kurita, Daisuke Kawahara, and Sadao Kurohashi. 2017. Neural joint model for transition-based Chinese syntactic analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*.

[4] https://github.com/fastnlp/fastNLP.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*.

Haonan Li, Zhisong Zhang, Yuqi Ju, and Hai Zhao. 2018. Neural character-level dependency parsing for Chinese. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2–7, 2018*.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for Chinese POS tagging and dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27–31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019*.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese word segmentation with bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? Word-based or character-based? In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25–26 July 2004, Barcelona, Spain*.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for Chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22–27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*.

Xian Qian and Yang Liu. 2012. Joint Chinese word segmentation, POS tagging and parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12–14, 2012, Jeju Island, Korea*.

Xipeng Qiu, Jiayi Zhao, and Xuanjing Huang. 2013. Joint Chinese word segmentation and POS tagging on heterogeneous annotated corpora with multiple task learning. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18–21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*.

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, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9–11, 2017*.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of the 2018*

*Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1–6, 2018, Volume 2 (Short Papers)*.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)*.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving Chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8–13, 2011*.

Zhiguo Wang, Chengqing Zong, and Nianwen Xue. 2013. A lattice-based framework for joint Chinese word segmentation, POS tagging and parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4–9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*.

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

Liner Yang, Meishan Zhang, Yang Liu, Maosong Sun, Nan Yu, and Guohong Fu. 2018. Joint POS tagging and dependence parsing with transition-based neural networks. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 26(8):1352–1358.

Meishan Zhang, Nan Yu, and Guohong Fu. 2018. A simple and effective neural model for joint word segmentation and POS tagging. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 26(9).

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level Chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22–27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15–20, 2008, Columbus, Ohio, USA*.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and pos-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9–11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18–21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019. Searching for effective neural extractive summarization: What works and what's next. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 1: Long Papers*.