A Unified Model for Joint Chinese Word Segmentation and Dependency Parsing

Hang Yan, Xipeng Qiu, Xuanjing Huang

***Abstract*—Chinese word segmentation and dependency parsing are two fundamental tasks for Chinese natural language processing. The dependency parsing is defined on word-level, therefore word segmentation is the precondition of dependency parsing, which makes dependency parsing suffers from error propagation. In this paper, we propose a unified model to integrate Chinese word segmentation and dependency parsing.**

Different from previous joint models, our proposed model is a graph-based model and more concise, which results in fewer efforts of feature engineering. Our joint model achieves better performance than previous joint models. Our joint model achieves the state-of-the-art results in both Chinese word segmentation and dependency parsing.

***Index Terms*—Chinese Word Segmentation, Dependency Parsing, Deep Learning, Biaffine Parsing**

I. Introduction

Unlike English, Chinese sentence consists of continuous characters and lacks obvious boundaries between Chinese words. Since words are usually regarded as the minimum semantic units, 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 on word-level. To parse a Chinese sentence, the process is usually the following pipeline way: word segmentation, POS tagging and dependency parsing.

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

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

(2) Knowledge Sharing. These three tasks (word segmentation, POS tagging and dependency parsing) are strongly related. The criterion of Chinese word segmentation also depends on the 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 way separately trains three models, each for a single task, and cannot fully exploit the shared knowledge among the three tasks.

H. Yan, X. Qiu and X. Huang are with the School of Computer Science and the Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai 200433, China (e-mail: 11300720199@fudan.edu.cn;xpqiu@fudan.edu.cn;xjhuang@fudan.edu.cn).

A traditional solution to this error propagation problem is to use joint models [2, 3, 1]. These previous joint models mainly adopted a transition-based parsing framework to integrate the word segmentation, POS tagging and dependency parsing. Based on the 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 to the word-level transition parsing, the 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, the approximate strategies like greedy search or beam search are adopted in practice. However, the approximate strategies do not, in general, produce an optimal solution. 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. [1] 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, the graph-based models have made great progress for dependency parsing [4, 5], which fully exploit the ability of the bidirectional long short-term memory network (BiLSTM) [6] and attention mechanism [7] 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 (MST) from these weighted arcs.

In this work, we propose a unified model for joint Chinese word segmentation and dependency parsing, which integrates these two tasks in one graph-based parsing model. Since the segmentation is character-level task and dependency is a word-level task, we first formulate these two tasks into character-level graph-based parsing framework. In detail, our model contains (1) a deep BiLSTM encoder, which is able to capture the long-term contextual features for each character, (2) a biaffine attentional scorer [5] , which unified predicts segmentation and dependency relations on character-level. Besides, unlike the previous joint models [2, 3, 1], our unified model does not depend on the POS tagging task.

In the experiments on two popular datasets, we obtain state-of-the-art scores on Chinese word segmentation and

dependency parsing.

In this paper, we claim three contributions as the following:

- To the best of our knowledge, this is the first graph-based method to integrate Chinese word segmentation and dependency parsing in a unified model. The proposed unified model is very concise and easily implemented.
- Compared to the previous transition-based joint models, our proposed model is a graph-based model, which results in fewer efforts of feature engineering. Besides, our model can deal with the labeled dependency parsing task, which is not easy for the transition-based joint models.
- In experiments on datasets CTB-5 and CTB-7, our model achieves the state-of-the-art scores in joint Chinese word segmentation and dependency parsing, even without the POS information.

II. 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.

A. Joint Segmentation and POS tagging

Since the 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 [8]. The cross-tag is composed of a word boundary part and a POS part, e.g., "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. [9], Chen et al. [10], Shao et al. [11] utilized neural models to alleviate the efforts of feature engineering.

Another line of the joint segmentation and POS tagging method is transition-based method [12, 13], in which the joint decoding process is regarded as a sequence of action predictions. Zhang et al. [14] used a simple yet effective sequence-to-sequence neural model to improve the performance of the transition-based method.

B. Joint POS tagging and Dependency Parsing

Since 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. [15] proposed a transition-based joint POS tagging and dependency parsing model and show that the joint approach improves the accuracies of these two tasks. Yang et al. [16] extended this model by neural models to alleviate the efforts of feature engineering.

Li et al. [17] unitized the graph-based model to jointly optimize POS tagging and dependency parsing in a unique model. They also propose an effective POS tag pruning method which can greatly improve the decoding efficiency.

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

C. Joint Segmentation, POS tagging and Dependency Parsing

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

Hatori et al. [2] first proposed a transition-based joint model for Chinese word segmentation, POS tagging and dependency parsing, which stated that dependency information improves the performances of word segmentation and POS tagging. Zhang et al. [3] expanded this work by using intra-character structures of words and find the intra-character dependencies are helpful in word segmentation and POS tagging. Zhang et al. [18] proposed joint segmentation, POS tagging and dependency re-ranking system. This system requires a base parser to generate some candidate parsing results. Kurita et al. [1] follow the work of [2, 3] and use the bidirectional LSTM (BiLSTM) to extract features with n-gram character string embeddings as input.

A related work is the full character-level neural dependency parser [19], 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 [20] proposed a joint inference model for word segmentation, POS tagging and constituency parsing. However, their model suffers from the decoding complexity due to the large combined search space. Wang et al. [21] firstly 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 were employed 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.

III. Proposed Model

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

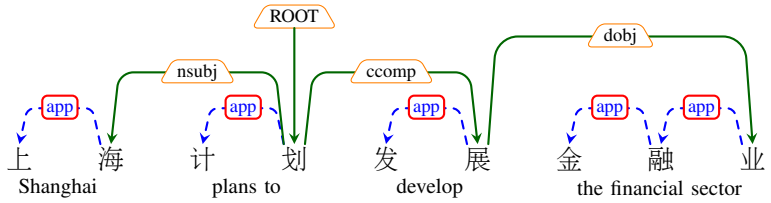

Fig. 1: Unified framework for joint Chinese word segmentation and dependency parsing. The green arc indicates the word-level dependency relation. The dashed blue arc with "app" indicates its connected characters belong to a word.

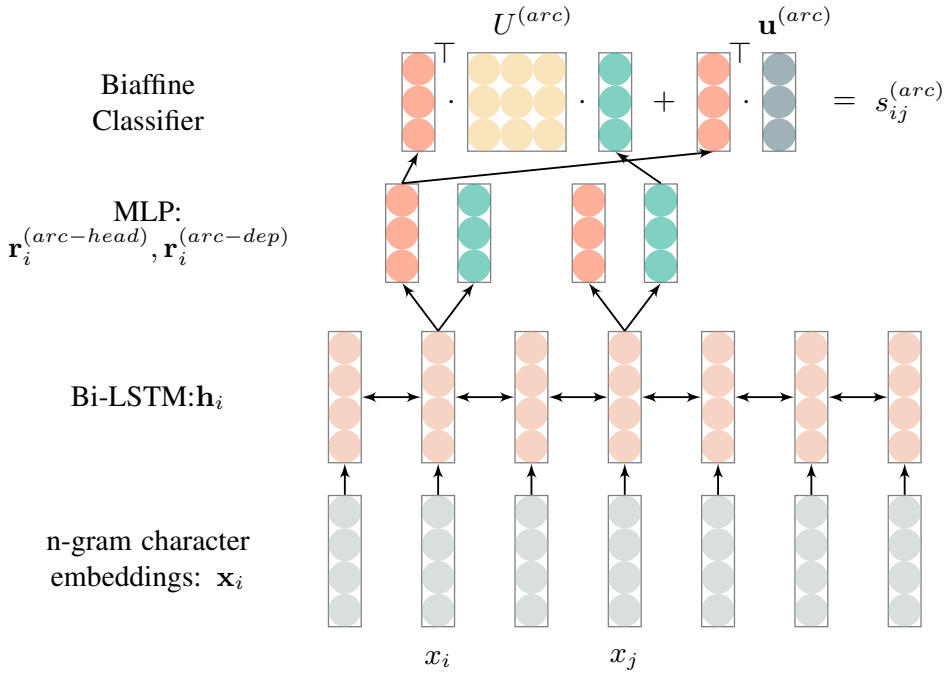

Fig. 2: Proposed joint model. For simplicity, we omit the prediction of arc label, which uses a different biaffine classifier.

Fig. 1 shows the proposed unified framework.

Firstly, we transform the word segmentation to a special arc prediction problem. For example, a Chinese word "金融业(financial sector)" has two intra-word dependent arcs: "金←融" and "融←业". Both the intra-word dependent arcs have the label "app". In this work, we simply define the last character in a word as the head character, and all the other characters depend on it.

Secondly, 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". Fig. 1 illustrate the unified framework for joint Chinese word segmentation and dependency parsing.

Thus, we can use a unified graph-based parsing model to conduct these two tasks. Our model contains two main components: (1) a deep BiLSTM encoder to extract the contextual features, which takes each character embedding of the given sentence as input and generates dense vectors, (2) a biaffine attentional scorer [5] which takes the hidden vectors for the given character pair as input and predict a label score vector.

Fig 2 illustrate the unified model for joint Chinese word segmentation and dependency parsing. The detailed description is as follows.

A. 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 Chinese word segmentation and POS tagging in previous studies [22, 23, 11, 14]. 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 $\mathbf{e}_i = \mathbf{e}_{c_i} \oplus \mathbf{e}_{c_i c_{i+1}} \oplus \mathbf{e}_{c_i c_{i+1} c_{i+2}}$, 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 bi-directional LSTM (BiLSTM) [6] 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

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

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 θ denotes all the parameters in BiLSTM layer.

B. Biaffine Layer

To predict the relations of each character pair, we employ the biaffine attention mechanism [5] to score their probability on the top of BiLSTM encoder. According to [5], biaffine attention is more effectively capable of measuring the relationship between two elementary units.

1) Unlabeled Arc Prediction: For the pair of the i-th and j-th characters, we first take the input of the BiLSTMs' hidden states $\mathbf{h}_i$, then feed them into an extension of bilinear transformation called a *biaffine function* to obtain the score for an arc from c_i (head) to c_j (dependent).

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

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

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

where MLP is a multi-layer perceptron. A weight matrix $U^{(arc)}$ determines the strength of a link from c_i to c_j while $\mathbf{u}^{(arc)}$ is used in the bias term, which controls the prior headedness of c_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 get 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 [5].

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 $c_i \rightarrow c_j$, 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.

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

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

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

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

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

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

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 $c_i \rightarrow c_j$ is determined according to $\mathbf{s}_{ij}^{(label)}$.

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

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" just occurs 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 tree based on the predicted character-level arc labels. The characters with continuous "app" are regarded as one word. And he predicted head of the last character of a word is viewed as this word's head. Since the predicted arc points to a character, we assign the word which contains this head character as the head word.

C. Models for Word Segmentation Only

The proposed unified model can be also used for Chinese word segmentation 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 which 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 character belongs to one word. Since 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.

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

TABLE I: Data statistics for CTB-5 and CTB-7

Dataset	Partition	♯sent	♯word	♯OOV
CTB-5	Training	18k	494k	-
	Develop	350	6.8k	553
	Test	348	8.0k	278
CTB-7	Training	31k	718k	-
	Develop	10k	237k	13k
	Test	10k	245k	13k

IV. Experiments

A. Datasets

We use the Penn Chinese Treebank 5.0 (CTB-5) and 7 (CTB-7) datasets to evaluate our models. For CTB-5, the training set is from sections 1∼270, 400∼931, and 1001∼1151, the development set is from section 301∼325, and the test set is from section 271∼300, this splitting was also adopted by [13, 3, 1]. For CTB-7, we use the splitting of [24, 3, 1]. The statistics of datasets are presented in Table I.

B. Measures

Following [2, 3, 18, 1], 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. Besides, we also report the standard metric for word-level dependency parsing, namely unlabeled attachment score (UAS) and labeled attachment score (LAS). We detail them in the following.

- $F1_{seg}$: F1 measure of Chinese word segmentation measure. This is the standard metric used in Chinese word segmentation task [25, 10]. The precision of Chinese word segmentation (denoted as P_{seg}) is calculated by the number of correctly segmented words versus the total number of segmented words. The recall of Chinese word segmentation (denoted as R_{seg}) is computed by the number of correctly segmented words versus the total number of golden words. Then we get $F1_{seg}$ by $F1_{seg} = 2 * P_{seg} * R_{seg}/(P_{seg} + R_{seg})$.
- $F1_{udep}$: F1 measure of unlabeled dependency parsing. Following [2, 3, 18, 1], 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 that the match between the head and dependent words must have the same label as the golden dependent-head pair. The precision and recall are calculated correspondingly. Since 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 $F1_{udep}$.
- UAS: Unlabeled attachment score. Because in the dependency parsing task, each word has only one head word, the percentage of words that have the correct head can be used as a metric to assess the performance. For our unified dependency parsing, this value equals to the value of the recall of unlabeled dependency parsing (R_{udep}).
- LAS: Labeled attachment score. This value measures the accuracy of dependency labels, and a dependency label is viewed as correct when its arc and its label are correct. For our unified dependency parsing, this value equals to the value of the recall of labeled dependency parsing (R_{ldep}).

The more detailed description of dependency parsing metrics can be found in [26].

C. Experimental Settings

a) Pretrained Embedding: Based on [11, 14], n-grams are of great benefit to Chinese word segmentation and POS tagging tasks, thus we use unigram, bigram and trigram embeddings for all of our character-based models. We first pretrain unigram, bigram and trigram embeddings on Chinese Wikipedia corpus by the method proposed in [27] 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 ..."; the trigram sequence is "abc bcd ...". For our word dependency parser, we use Tencent's pretrained word embeddings [28]. Because the Tencent's pretrained word embedding dimension is 200, we set both pretrained and random word embedding size as 200 for all of our word dependency parsing models. All pretrained embeddings are fixed during our experiments. In addition to the fixed pretrained embeddings, we also randomly initialize embeddings, and element-wise add the pretrained and random embeddings before other procedures.

b) Hyper-parameters: The development set is used for parameter tuning. We generally follow the hyperparameters

TABLE II: Hyper-Parameter Settings

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	$.75^{t/5000}$
β_1, β_2	0.9
Max epochs	100

chosen in [5]. Specifically, we use with 400 units for each BiLSTM layer. The recurrent dropout rates are all 0.33.

The model is trained with the Adam algorithm [29] to minimize the sum of the cross-entropy of arc predictions and label predictions. All models are trained for 100 epochs, after each training epoch, we test the model on the dev set, and models with the highest $F1_{udep}$ in dev set are kept. Detail hyperparameters can be found in Table II.

D. Proposed Models

In this part, we introduce the settings for our proposed joint models. Based on whether the model uses dependency parsing labels, we divide our models into two kinds. We enumerate them as follows.

- **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 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-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 detailed model is described in Section III-C.

For the above three models, all the settings are the same, except for the distinction of the label set.

E. Comparison with the Previous Joint Models

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

As presented in Table III, our joint model (the penultimate row) outpaces previous methods with a large margin in both Chinese word segmentation and dependency parsing, even without the local parsing features which were extensively used in previous transition-based joint work [2, 3, 18, 1]. Another difference between our joint models and previous works is the combination of POS tags, the previous models all used POS task as one componential task. Despite the lack of POS tag information, our models still achieve much better results. However, according to [5], 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 the last two rows of Table III, the joint model with labeled dependency parsing outperforms its counterpart in both Chinese word segmentation 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 better Chinese word segmentation. and the more accurate arc label can reduce the error of word segmentation. The second one is that label information between two characters will give extra supervision for the search of head characters. 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 reason why labeled dependency parsing is conducive to the Chinese word segmentation will be also analyzed in Section IV-F.

F. Chinese Word Segmentation

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

Since most of the state-of-art Chinese word segmentation 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, S represents the word has only one character. We compare our model with these state-of-art methods.

- LSTM+MLP with $\{B, M, E, S\}$ tags. Following [30], we tried to do Chinese word segmentation without conditional random field (CRF). After BiLSTM, the hidden states of each character are further forward into a multi

TABLE III: Main Results

Models	CTB-5		CTB-7	
	$F1_{seg}$	$F1_{udep}$	$F1_{seg}$	$F1_{udep}$
Hatori et al. [2]	97.75	81.56	95.42	73.58
Zhang et al. [3] STD	97.67	81.63	95.53	75.63
Zhang et al. [3] EAG	97.76	81.70	95.39	75.56
Zhang et al. [18]	98.04	82.01	-	-
Kurita et al. [1]	98.37	81.42	95.86	74.04
Joint-binary	98.45	86.72	96.57	81.34
Joint-multi	**98.47**	**87.24**	**96.66**	**81.98**

[1] STD and EAG in [3] denote the arc-standard and the arc-eager models.
[2] $F1_{seg}$ and $F1_{udep}$ are two metrics to evaluate the unified dependency parsing.

TABLE IV: Results of Chinese word segmentation.

Models	Tag Set	CTB-5			CTB-7		
		$F1_{seg}$	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
LSTM+CRF	$\{B, M, E, S\}$	98.48	98.33	98.68	96.46	96.45	96.47
LSTM+MLP	$\{app, seg\}$	98.40	98.14	98.66	96.41	96.53	96.29
Joint-SegOnly	$\{app, seg\}$	**98.50**	**98.30**	98.71	96.50	96.67	96.34
Joint-Binary	$\{app, dep\}$	98.45	98.16	98.74	96.57	96.66	96.49
Joint-Multi	$\{app, dep_1, \cdots, dep_K\}$	98.47	98.14	**98.80**	**96.66**	**96.71**	**96.61**

[1] The upper part refers the state-of-the-art models based on sequence labeling.
[2] The lower part refers our proposed joint models which are detailed in section IV-D. The proposed joint models achieve near or better $F1_{seg}$ than models trained only on Chinese word segmentation.
[2] $F1_{seg}$ R_{seg} and R_{seg} are the F1, precision and recall of Chinese word segmentation respectively.

layer perceptron (MLP), so that every character can output a probability distribution over the label set. 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 use conditional random field (CRF) after the MLP [31, 10].
- LSTM+MLP with $\{app, seg\}$ tags. The segmentation of a Chinese sentence can be represented by a sequence of $\{app, seg\}$, where app represents the next character and this character belongs to the same word, and seg represents this character is the last character of a word. Therefore, Chinese word segmentation 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 the above models have the same encoder as our joint models, they differ from each other in their way of decoding and the tag set. The number of LSTM layers is 2 or 3 and the BiLSTM hidden size is 200.

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

Different to the competitor sequence labeling model (LSTM+MLP with $\{app, seg\}$ tag set), our joint-SegOnly mode uses the biaffine to model the interaction between the two adjacent characters near the boundary and achieves better performances on both datasets. The empirical results in two datasets suggest that modeling the interaction between two consecutive characters are helpful to Chinese word segmentation.

The lower part of Table IV shows the segmentation evaluation of all the proposed joint models. Joint training Chinese word segmentation and dependency parsing achieves comparable or better Chinese word segmentation than training Chinese word segmentation alone. Although head prediction is not directly related to Chinese word segmentation, the head character can only be the end of a word, therefore combination between Chinese word segmentation 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.

Another noticeable phenomenon from the lower part of Table IV is that labeled dependency parsing brings more benefits to Chinese word segmentation. We assume there are two explanations. Firstly, the incorporation of dependency parsing labels allays the label imbalance between "seg" and "app" labels. For datasets CTB-5 and CTB-7, their ratios in the training data are 61:39 and 62:38 respectively. However, when combining with dependency labels, the dominant "seg" part will be divided into several different labels. Secondly, the extra supervision from dependency parsing labels is informative for word segmentation.

G. 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 get segmentation results, then apply the word-based biaffine parser to parse the segmented sentence. The word-level biaffine parser is the same with [5] but without POS tags. Just like the unified parsing metric, for a dependent-head word pair, only when both head and dependent words are correct, this pair can be viewed as a right one.

From Table V, it obviously shows that in both CTB-5 and CTB-7, the Joint-Multi model consistently outperforms the pipeline model in $F1_{udep}$, UAS, $F1_{ldep}$ and LAS. In CTB-7, owing to higher increasement in $F1_{seg}$, the increasement of parsing metrics compared to the pipeline model is much larger than that in CTB-5. Although the $F1_{seg}$ difference between the Joint-Multi model and the pipeline model is only -0.03 and +0.15 in CTB-5 and CTB-7 respectively, but the $F1_{udep}$ of the joint-multi is higher than the pipeline model by +0.74 and +1.36 respectively, we believe this indicates the better resistance to error propagation of the Joint-Multi model.

H. 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, one is either head or dependent (or both) wrongly segmented, the other is the wrong choice on the head word. The ratio of these two mistakes is presented in Table VI. For the Joint-Multi model, more mistakes caused by segmentation in CTB-7 is coherent with our observation that CTB-7 bears lower Chinese word segmentation performance. Based on our error analysis, the wrong prediction of head word accounts for most of the errors, therefore further joint models address head prediction error problem might get more gain on performance.

Additionally, although from Table V the distinction of $F1_{seg}$ between the Joint-Multi model and the pipeline model is around +0.1% in average, the difference between the Head-wrong is more than around +0.72% in average. We think this is caused by the pipeline model is more sensitive to word segmentation errors.

I. Ablation Study

As our model uses various n-gram pretrained embeddings, we explore the influence of these pretrained embeddings. The second row in Table VII shows the results of the Joint-Multi model without pretrained embeddings, it is clear that pre-trained embeddings are important for both word segmentation and dependency parsing.

We also tried to remove the bigram and trigram. Results are illustrated in the third row of Table VII. 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. For Chinese word segmentation, the model without pretrained embeddings gets superior performance than without bigram and trigram feature. While for all dependency parsing related metrics, the model with pretrained character embedding gets better performance. We assume the n-gram features are important to Chinese word segmentation. But for the dependency parsing task, the relation between two characters are more beneficial, when pretraining embeddings are combined, the model can exploit the relationship encoded in the pretrained embeddings. Additionally, even though the third row has inferior $F1_{seg}$ (in average 0.16% lower than the second row), it still achieves much better $F1_{udep}$ (in average 0.95% higher than the second row). We believe this is proof that joint Chinese word segmentation and dependency parsing is resistant to error propagation.

V. Conclusion and Future Work

In this paper, we propose a unified model for joint Chinese word segmentation and dependency parsing. Different from the previous joint models, our proposed model is a graph-based model and 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 in a large margin. The empirical results in CTB-5 and CTB-7 show that the dependency parsing task is also beneficial to Chinese word segmentation. Besides, labeled dependency parsing not only is good for Chinese word segmentation, but also avails the dependency parsing head prediction. And our joint model with labels unveils the state-of-the-art results both in Chinese word segmentation and dependency parsing.

Apart from the good performance, the comparison between our joint model and the pipeline model shows great potentialities for unified Chinese dependency parsing. Our proposed method not merely outpaces the pipeline model, but also avoids the preparation for pretrained word embeddings which 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 [3] more reasonable intra-word dependent structure might further boost the performance of all tasks.

TABLE V: Comparison with the pipeline model.

Models	CTB-5					CTB-7				
	$F1_{seg}$	$F1_{udep}$	UAS	$F1_{ldep}$	LAS	$F1_{seg}$	$F1_{udep}$	UAS	$F1_{ldep}$	LAS
Biaffine[§]	-	-	88.81	-	85.63	-	-	86.06	-	81.33
Pipeline	**98.50**[†]	86.50	86.71	83.46	83.67	96.51[†]	80.62	80.49	76.58	76.46
Joint-Multi	98.47	**87.24**	**87.58**	**84.54**	**84.89**	**96.66**	**81.98**	**81.96**	**77.95**	**77.93**

[1] The pipeline model first uses the Joint-SegOnly model to segment the sentence, then uses the word-level biaffine parser to get the parsing result, whose performance is displayed in the first row.
[†] The results are evaluated on the gold-segmented sentences.
[§] The results are from the Joint-SegOnly model.

TABLE VI: Error Analysis of Unlabeled Dependency Parsing.

Model	CTB-5			CTB-7		
	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%
Joint-Multi	86.90%	3.43%	9.67%	82.00%	6.75%	11.25%

[1] 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 VII: Ablation experiments.

Models	CTB-5					CTB-7				
	$F1_{seg}$	$F1_{udep}$	UAS	$F1_{ldep}$	LAS	$F1_{seg}$	$F1_{udep}$	UAS	$F1_{ldep}$	LAS
Joint-Multi	98.47	87.24	87.58	84.54	84.89	96.66	81.98	81.96	77.95	77.93
-pretrained	97.72	82.56	82.70	79.8	70.93	95.52	76.35	76.22	72.16	72.04
-n-gram	97.72	83.44	83.60	80.24	80.41	95.21	77.37	77.11	72.94	72.69

[1] The '-pretrained' means the model is trained without the pretrained embeddings.
[2] The '-n-gram' means the model is trained by removing the bigram and trigram embeddings, only randomly initialized and pretrained character embeddings are used.

VI. Acknowledgements

This work was supported by the National Key Research and Development Program of China [grant numbers 2017YFB1002104], and the National Natural Science Foundation of China [grant numbers 61672162, 61751201].

References

[1] S. Kurita, D. Kawahara, and S. Kurohashi, "Neural joint model for transition-based Chinese syntactic analysis," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 1204–1214.

[2] J. Hatori, T. Matsuzaki, Y. Miyao, and J. Tsujii, "Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 2012, pp. 1045–1053.

[3] M. Zhang, Y. Zhang, W. Che, and T. Liu, "Character-level Chinese dependency parsing," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2014, pp. 1326–1336.

[4] E. Kiperwasser and Y. Goldberg, "Simple and accurate dependency parsing using bidirectional lstm feature representations," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 313–327, 2016.

[5] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," *arXiv preprint arXiv:1611.01734*, 2016.

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

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

[8] H. Ng and J. Low, "Chinese part-of-speech tagging: one-at-a-time or all-at-once? word-based or character-based," in *Proceedings of EMNLP*, vol. 4, 2004.

[9] X. Zheng, H. Chen, and T. Xu, "Deep learning for chinese word segmentation and pos tagging," in *Proceedings*

of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 647–657.

[10] X. Chen, X. Qiu, and X. Huang, "A feature-enriched neural model for joint Chinese word segmentation and part-of-speech tagging," in *IJCAI*, 2017.

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

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

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

[14] M. Zhang, N. Yu, and G. Fu, "A simple and effective neural model for joint word segmentation and POS tagging," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 9, pp. 1528–1538, 2018.

[15] J. Hatori, T. Matsuzaki, Y. Miyao, and J. Tsujii, "Incremental joint POS tagging and dependency parsing in chinese," in *Proceedings of 5th international joint conference on natural language processing*, 2011, pp. 1216–1224.

[16] L. Yang, M. Zhang, Y. Liu, M. Sun, N. Yu, and G. Fu, "Joint POS tagging and dependence parsing with transition-based neural networks," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 8, pp. 1352–1358, 2018.

[17] Z. Li, M. Zhang, W. Che, T. Liu, W. Chen, and H. Li, "Joint models for chinese POS tagging and dependency parsing," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 1180–1191.

[18] Y. Zhang, C. Li, R. Barzilay, and K. Darwish, "Randomized greedy inference for joint segmentation, pos tagging and dependency parsing," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 42–52.

[19] H. Li, Z. Zhang, Y. Ju, and H. Zhao, "Neural character-level dependency parsing for Chinese," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[20] X. Qian and Y. Liu, "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*. Association for Computational Linguistics, 2012, pp. 501–511.

[21] Z. Wang, C. Zong, and N. Xue, "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 (Volume 2: Short Papers)*, vol. 2, 2013, pp. 623–627.

[22] W. Pei, T. Ge, and B. Chang, "Max-margin tensor neural network for Chinese word segmentation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2014, pp. 293–303.

[23] X. Chen, X. Qiu, C. Zhu, P. Liu, and X. Huang, "Long Short-Term Memory Neural Networks for Chinese Word Segmentation." in *EMNLP*, 2015, pp. 1197–1206.

[24] Y. Wang, J. Kazama, Y. Tsuruoka, W. Chen, Y. Zhang, and K. Torisawa, "Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, 2011, pp. 309–317.

[25] X. Qiu, J. Zhao, and X. Huang, "Joint chinese word segmentation and POS tagging on heterogeneous annotated corpora with multiple task learning." in *EMNLP*, 2013, pp. 658–668.

[26] S. Kübler, R. McDonald, and J. Nivre, "Dependency parsing," *Synthesis Lectures on Human Language Technologies*, vol. 1, no. 1, pp. 1–127, 2009.

[27] W. Ling, C. Dyer, A. W. Black, and I. Trancoso, "Two/too simple adaptations of word2vec for syntax problems," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1299–1304.

[28] Y. Song, S. Shi, J. Li, and H. Zhang, "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, Volume 2 (Short Papers)*, vol. 2, 2018, pp. 175–180.

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

[30] J. Ma, K. Ganchev, and D. Weiss, "State-of-the-art Chinese word segmentation with Bi-LSTMs," *arXiv preprint arXiv:1808.06511*, 2018.

[31] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.