Deep Graph-Based Character-Level Chinese Dependency Parsing

Linzhi Wu and Meishan Zhang

***Abstract*—Character-level Chinese dependency parsing has been a concern of several studies that naturally handle word segmentation, POS (Part of Speech) tagging and dependency parsing jointly in an end-to-end way. Previous work mostly concentrates on a transition-based framework for this task because of its easy adaption, which is extremely important when feature representation relies heavily on the decoding strategy, particularly under the traditional statistical setting. Recently, on the one hand, sophisticated deep neural networks and deep contextualized word representations have greatly weakened the dependence between feature representation and decoding. On the other hand, (first-order) graph-based models, especially the biaffine parsers, are straightforward for dependency parsing, and meanwhile they can yield competitive parsing performance. In this paper, we make a comprehensive investigation of the deep graph-based character-level dependency parsing for Chinese. We start from an extension of a standard graph-based biaffine parser, and then exploit Chinese BERT as well as our improved encoders based on transformers to enhance the character-level dependency parsing model. We conduct a series of experiments on the Chinese benchmark datasets, showing the performances of various graph-based character-level models and analyzing the advantages of the character-level dependency parsing under the deep neural setting.**

***Index Terms*—Character-level chinese parsing, deep neural networks, dependency parsing, graph-based model.**

I. Introduction

DEPENDENCY parsing is a fundamental task in the natural language processing (NLP) community, which has been studied intensively for decades [1]–[9]. Given an input sentence, the task aims to disclose the syntactic or semantic relationships between the sentential words. Since Chinese sentences have no explicit boundaries between words, a prerequisite word segmentation is usually assumed to align with the current state-of-the-art word-level dependency parsing models [9]–[12]. In addition, as POS tags are valuable word-level feature source, conventional Chinese dependency parsing usually involves three successive steps: word segmentation, POS tagging and word-level dependency parsing.

Manuscript received May 28, 2020; revised December 2, 2020 and March 6, 2021; accepted March 6, 2021. Date of publication March 18, 2021; date of current version April 8, 2021. This work was supported by the National Natural Science Foundation of China under Grants 61602160 and 61672211. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Eric Fosler-Lussier. *(Corresponding author: Meishan Zhang.)*

The authors are with the School of New Media and Communication, Tianjin University, Tianjin 300072, China (e-mail: tjuwlz2020@163.com; mason.zms@gmail.com).

Digital Object Identifier 10.1109/TASLP.2021.3067212

The architecture mentioned above handles Chinese dependency parsing in a pipeline manner, which may suffer from the error propagation problem, where early-step errors may influence the future-step analysis. Character-level dependency parsing has been widely adopted to perform Chinese dependency parsing jointly [13]–[16]. By using the well-defined inner-word character dependencies, we can extend word-level dependency trees into the character-level ones naturally. Fig. 1 shows an example of a character-level dependency tree, where all the inner-word characters are headed to their right-adjacent characters with a predefined dependency label (e.g., #in). With this formalization, Chinese dependency parsing can be conducted at the character level. Moreover, word segmentation, POS tagging and dependency parsing can be achieved by a single joint model, leading to an end-to-end solution for Chinese dependency parsing.

Previously, character-level Chinese dependency parsing has been widely addressed by transition-based models [13]–[16], since the framework is highly flexible for decoding extension, and meanwhile, it is easy to integrate arbitrary features, which are very important to character-level parsing in Chinese, especially under the traditional statistical models. Recently, graph-based biaffine dependency parsing has received great attention [9], [11], [12] on account of its competitive parsing performance, and the encoder-decoder architecture has made the feature representation less dependent on the decoding process [9]. In reality, decoder-independent feature representations can be further enhanced by the deep contextualized word representations such as ELMo and BERT [17], [18], leading to a much powerful encoder. Thus, it is expectable to handle character-level Chinese dependency parsing based on the graph-based methods under the neural setting.

In this work, we make a comprehensive study of graph-based character-level Chinese dependency parsing with deep neural networks. We follow the work of biaffine dependency parsing [9], adapting it to our character-level parsing. The extension is straightforward. We conduct the biaffine operations over Chinese characters directly for character-level dependency link prediction, with no change in either the training or the inference. Following Zhang *et al.* (2017) [13], our character-level model also performs word segmentation and POS tagging jointly, which is achieved by character-level sequence labeling, where the labels indicate the joint word boundary and POS tagging information. All the subtasks are organized by a multi-task learning (MTL) framework [19].

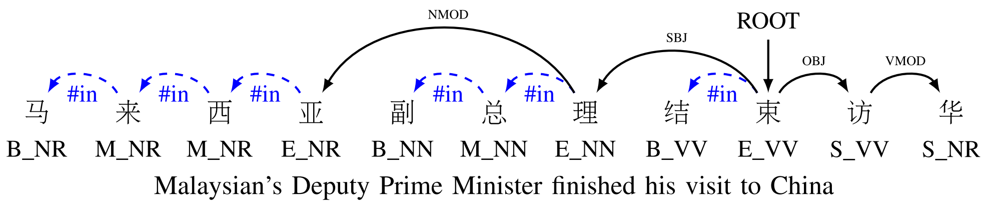

NMOD
SBJ
ROOT
OBJ
VMOD
#in
马 来 西 亚 副 总 理 结 束 访 华
B_NR M_NR M_NR E_NR B_NN M_NN E_NN B_VV E_VV S_VV S_NR
Malaysian's Deputy Prime Minister finished his visit to China

Fig. 1. An example of a character-level Chinese dependency tree based on "********** (Malaysian's Deputy Prime Minister finished his visit to China)". The inner-word dependencies in the example are left-branching, and actually we can also exploit the manually-annotated dependencies by Zhang *et al.* (2017) [13].

As for the encoder part, we enhance the character-level parsing model in two ways. First, we investigate Chinese BERT to obtain stronger contextualized character representations, which have been demonstrated very effective on many related tasks [11], [12], [20]. BERT can be used by way of either feature-based (i.e., frozen parameters) or fine-tuning, both of which are important for NLP modeling. We study both settings, and in particular, exploit the adapter module [21], [22] to make the performance of the feature-based strategy on par with fine-tuning. Second, we suggest a modified version of the standard Transformer [23] by integrating relative position embeddings (i.e., RPE-Transformer) to enhance the modeling of short-term connections, which is important to our joint task.

We conduct a series of experiments on three general Chinese benchmark datasets. Experimental results show that the graph-based character-level parsing model can achieve very competitive performance on all the subtasks, including word segmentation, POS tagging and dependency parsing. Both the BERT representations and our proposed RPE-Transformer encoder can bring further improvements to our models. Our proposed model can be applicable for the manually-annotated inner-word word dependencies presented in Zhang *et al.* (2017) [13] naturally. However, the experimental results show that no significant differences can be observed after incorporating them. We compare our joint models with the corresponding pipelines and show that the end-to-end joint framework can gain better performance. Extensive analysis studies are conducted in detail, aiming to understand the character-level parsing model comprehensively.

In summary, our major contributions are listed as follows:

- We present a comprehensive study for deep graph-based character-level Chinese dependency parsing, including investigations by using various word representations and inner-word dependencies as well as comparisons with different pipelines. To our knowledge, it is the first work to examine BERT, gold-standard inner-word structures, and their corresponding pipelines for this task under the deep neural setting.
- We propose a novel RPE-Transformer to improve our joint models that use the feature-based method to utilize external pre-trained character representations. This achievement is highly meaningful because the feature-based method is very parameter-efficient (e.g., a static BERT could be preserved by sharing across different NLP models), thus our final feature-based model would be more desirable in real considerations since a comparable parsing performance can also be achieved.

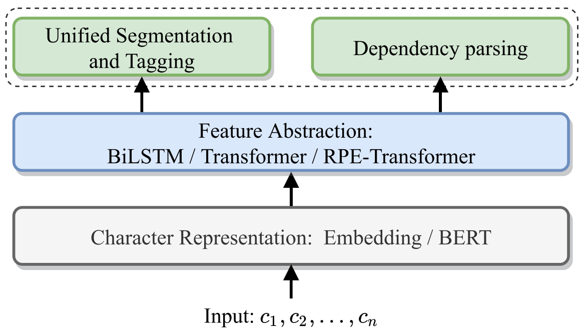

Unified Segmentation and Tagging
Dependency parsing
Feature Abstraction: BiLSTM / Transformer / RPE-Transformer
Character Representation: Embedding / BERT
Input: $c_1, c_2, \ldots, c_n$

Fig. 2. The proposed framework for joint Chinese word segmentation, POS tagging and dependency parsing.

All code has been released publicly available under Apache License 2.0 for research purposes.[1]

II. The Proposed Model

Our graph-based character-level Chinese dependency parsing model is directly adapted from the word-level biaffine parsing model proposed by Dozat and Manning (2017) [9]. Fig. 2 shows the overall model architecture, which is divided into three parts: (1) character representation, (2) feature abstraction, and (3) decoding, where the first two parts form the encoder, and the third is the decoder. Based on this framework, we can process word segmentation, POS tagging and dependency parsing in a single model, where Chinese characters are adopted as the basic processing units.

The encoder is highly similar to the word-level dependency parsing while the decoder differs significantly because word segmentation and POS tagging are integrated, which should be tackled carefully to achieve good performance as a whole. As shown in Fig. 2, besides the standard biaffine dependency parsing over characters, we exploit one extra component to handle word segmentation and POS tagging by a unified sequence labeling framework, namely unified segmentation and tagging, which assigns each character to a joint label that unites word boundary tags $\{\mathrm{B}, \mathrm{M}, \mathrm{E}, \mathrm{S}\}$[2] and POS tags T as a whole [24], i.e.

[1]https://github.com/LindgeW/JointCWPDParser

[2]BMES are widely used word boundary tags, where BME indicates the beginning, middle and end characters of a word, respectively, and S represents a single-character word.

$\{\mathrm{B}, \mathrm{M}, \mathrm{E}, \mathrm{S}\} \times T$, where $\times$ denotes the Cartesian product. Finally, we organize the two components in a MTL framework [19] for decoding.

A. *Character Representation*

We take Chinese characters as the basic input for our character-level dependency parser. Given an input sequence with n characters $c_1 \cdots c_n$, we mainly investigate two kinds of character-level representations, one being the baseline embeddings, and the other one being the BERT representations. In this work, we do not investigate the ELMo representations [17] for our tasks, which may result in performance between the embeddings and BERT [11].

Embedding. We use embeddings of unigram and bigram characters to obtain the hidden vector representation at each position, following previous neural character-level Chinese processing models such as word segmentation, and joint word segmentation and POS tagging [24], [25]. Concretely, for each position $i \in [1, n]$, we make embeddings of c_i and $c_i c_{i+1}$, resulting in the character-level representation as follows:

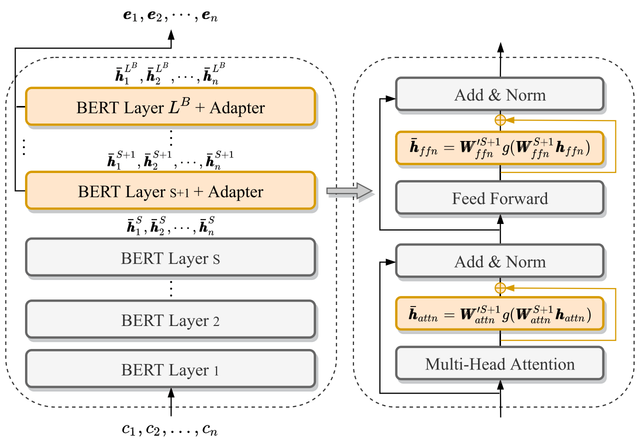
$$\begin{aligned} \boldsymbol{e}_i = & \left(\boldsymbol{E}_{\text{fix}}(c_i) + \boldsymbol{E}_{\text{tune}}(c_i)\right) \\ & \oplus \left(\boldsymbol{E}_{\text{fix}}(c_i c_{i+1}) + \boldsymbol{E}_{\text{tune}}(c_i c_{i+1})\right), \end{aligned} \tag{1}$$

where $\boldsymbol{E}_{\text{fix}}$ is an embedding matrix with pre-trained vectors which is frozen in our models, $\boldsymbol{E}_{\text{tune}}$ is a randomly initialized embedding matrix and would be fine-tuned along with the training, $\boldsymbol{E}(\cdot)$ indicates the matrix looking up function, $\oplus$ denotes vectorial concatenation, and $\boldsymbol{e}_i$ is our desired output.

BERT (tuned). BERT is a powerful language representation model, which has achieved the state-of-the-art results on a wide range of NLP tasks [18]. It accepts a full sentence as input, outputting a sequence of contextualized word representations based on a well-pretrained bidirectional Transformer. For the standard BERT of Chinese, characters are basic processing units instead of words, which perfectly fits with our setting. First, following the standard settings of Devlin *et al.* (2019) [18], we use the last-layer vectorial output for our character-level dependency parser and fine-tune the BERT parameters along with our task objective, which has been demonstrated effective in previous work [18], [26].

BERT (frozen). Although fine-tuning BERT can achieve remarkable performances for a number of tasks [18], [27], this scheme may suffer from the parameter inefficiency problem, where a newly-trained model would introduce a new copy of BERT weights (i.e., consuming about 110 M parameters). This may lead to great inconveniences in real scenarios which involve multiple NLP tasks and model ensembles, since each model keeps a different copy of BERT weights. Thus it is highly meaningful to study the feature-based method which freezes BERT parameters during training.[3] In this way, we can preserve a shared BERT across different NLP models. Thus, in this work, we also investigate the performance of BERT (frozen) for character-level parsing in Chinese.

[3]Our embedding-based joint model also leverages the feature-based method naturally to integrate the external pre-trained word embeddings.



$\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_n$
$\bar{\boldsymbol{h}}_1^{L^B}, \bar{\boldsymbol{h}}_2^{L^B}, \cdots, \bar{\boldsymbol{h}}_n^{L^B}$
BERT Layer L^B + Adapter
$\bar{\boldsymbol{h}}_1^{S+1}, \bar{\boldsymbol{h}}_2^{S+1}, \cdots, \bar{\boldsymbol{h}}_n^{S+1}$
BERT Layer S+1 + Adapter
$\bar{\boldsymbol{h}}_1^{S}, \bar{\boldsymbol{h}}_2^{S}, \cdots, \bar{\boldsymbol{h}}_n^{S}$
BERT Layer S
BERT Layer 2
BERT Layer 1
$c_1, c_2, \ldots, c_n$
Add & Norm
$\bar{\boldsymbol{h}}_{ffn} = \boldsymbol{W}'^{S+1}_{ffn} g(\boldsymbol{W}^{S+1}_{ffn} \boldsymbol{h}_{ffn})$
Feed Forward
Add & Norm
$\bar{\boldsymbol{h}}_{attn} = \boldsymbol{W}'^{S+1}_{attn} g(\boldsymbol{W}^{S+1}_{attn} \boldsymbol{h}_{attn})$
Multi-Head Attention

Fig. 3. Illustration of character representations based on BERT (frozen). **Left**: The overall architecture. **Right**: The inner structure of the BERT layer S+1 with adapter, where the brown part is the adapter layers, each of which involves a combination of down-projection (i.e., W'^{S+1}_{*} and $\mathrm{dim}_{\text{out}} \ll \mathrm{dim}_{\text{in}}$), activation (i.e., $g = \mathrm{GeLU}$), up-projection (i.e., W'^{S+1}_{*}, restoring to the BERT dim) and skip-connection.

Our preliminary experiments show that direct feature extraction from BERT would lead to significant decreases in the overall parsing performance. To reduce the gap between fine-tuning and freezing, we exploit the adapter technique [21], [22], applying it to the last few layers of BERT, and then aggregate the outputs of these layers with weighed summation to obtain the final character-level representations. Fig. 3 presents the feature extraction method. The overall process can be formalized as follows:

$$\begin{aligned} & \bar{\boldsymbol{h}}_1^{S} \cdots \bar{\boldsymbol{h}}_n^{S} = \mathrm{BERT}^{:S}\left(c_1 \cdots c_n\right) \\ & \bar{\boldsymbol{h}}_1^{S+1:L^{\mathrm{B}}} \cdots \bar{\boldsymbol{h}}_n^{S+1:L^{\mathrm{B}}} = \text{BERT-AD}^{S+1:L^{\mathrm{B}}}\left(c_1 \cdots c_n\right) \\ & \boldsymbol{e}_i = \sum_{l=S+1}^{L^{\mathrm{B}}} \bar{\boldsymbol{\lambda}}_l \bar{\boldsymbol{h}}_i^{l}, i \in [1, n], \end{aligned} \tag{2}$$

where $\mathrm{BERT}^{:S}$ indicates the vector calculation following the standard BERT module in the bottom layers, $\text{BERT-AD}^{S+1:L^{\mathrm{B}}}$ denotes the adapter module is inserted starting from the $\{S+1\}$th layer of BERT to the last layer, and $\boldsymbol{e}_i$ is our desired output of character representation. We only offer a brief depiction of the adapter module in Fig. 3, and more detailed descriptions can be found in Houlsby *et al.* (2019) [22]. Note that during training, we only fine tune the weights of the adapter modules, the layer normalization parameters and $\bar{\boldsymbol{\lambda}}$, keeping all other BERT parameters fixed.

B. *Feature Abstraction*

Here we refer to the feature abstraction as one further module to produce high-level shared features for our joint task based on the input word-level representations. This part can be skipped when tunable BERT is exploited as the character-level representations because BERT can learn task-specific high-level features directly by adjusting a large quantity of parameters itself. For other character-level representations that are regarded as feature-based, an additional feature abstraction module is very necessary to achieve competitive performance. In this work, we investigate three different feature abstraction strategies, which are built using BiLSTM, Transformer, and an improved Transformer, respectively.

BiLSTM. The BiLSTM encoder is exploited in the original (word-level) biaffine dependency parsing [9], which is used as our baseline encoder. Formally, given the character-level representations $\boldsymbol{e}_1 \cdots \boldsymbol{e}_n$, we firstly utilize a multi-layer BiLSTM module to obtain the encoder outputs:

$$\boldsymbol{h}_1^{1:L^{\text{E}}} \cdots \boldsymbol{h}_n^{1:L^{\text{E}}} = \text{BiLSTM}^{L^{\text{E}}}(\boldsymbol{e}_1 \cdots \boldsymbol{e}_n), \tag{3}$$

where L^{E} is the number of BiLSTM layers, and then we dump the outputs of all encoder layers for future decoding, which can also be applied to other kinds of encoders universally.

Transformer. We refer to the Transformer as the basic processing module of the encoder part mentioned in Vaswani *et al.* (2017) [23], also as the backbone of a single BERT layer as depicted in Fig. 3, which involves a combination of multi-head self-attention, layer normalization and feed-forward sublayers. Given the input vectors derived from the character representations $\boldsymbol{e}_1 \cdots \boldsymbol{e}_n$, we first add them with the sinusoidal positional embeddings [23], and then let them go through several Transformer encoder layers:

$$\begin{aligned} \boldsymbol{h}_i^0 &= \boldsymbol{e}_i + \boldsymbol{e}'_i \\ \boldsymbol{h}_1^l \cdots \boldsymbol{h}_n^l &= \text{Transformer}(\boldsymbol{h}_1^{l-1} \cdots \boldsymbol{h}_n^{l-1}) \end{aligned} \tag{4}$$

where $\boldsymbol{e}'_i$ denotes the position embeddings, and finally we can obtain the encoder outputs $\boldsymbol{h}_1^{1:L^{\text{E}}} \cdots \boldsymbol{h}_n^{1:L^{\text{E}}}$ by a L^{E}-layer Transformer (i.e., $l \in [1, L^{\text{E}}]$).

RPE-Transformer. According to our preliminary experiments, we find that the Transformer encoder is hard to bring improved performance consistently. We can observe the similar findings from Li *et al.* (2019) [11] who investigate the performance of the Transformer for word-level dependency parsing. One possible reason might be due to that the attention mechanism inside the multi-head self-attention sublayers treats the representations of different positions equally. Compared with BiLSTM, the short-distance connections in the standard Transformer have been weakened significantly. Therefore, we attempt to improve the Transformer by incorporating the relative position embeddings (RPE) into the multi-head self-attention components, which is mainly motivated by Shaw *et al.* (2018) [28] and Dai *et al.* (2019) [29], enhancing the ability of short-term connection awareness according to the objective of our task.

The key attention mechanism in the Transformer can be depicted in Fig. 4, which illustrates the computation flow of single-head attention. Multi-head attention is a simple extension by separating one single input vector into several parts, each of which executes single-head attention operation individually, and then concatenating the outputs of these parts as a whole. Assuming the input vector of the self-attention sublayer is $\boldsymbol{x}_{1:n} = \boldsymbol{x}_1 \cdots \boldsymbol{x}_n$, the attention scores are calculated by the formula shown in Fig. 4, the input vector $\boldsymbol{x}$ is separately projected

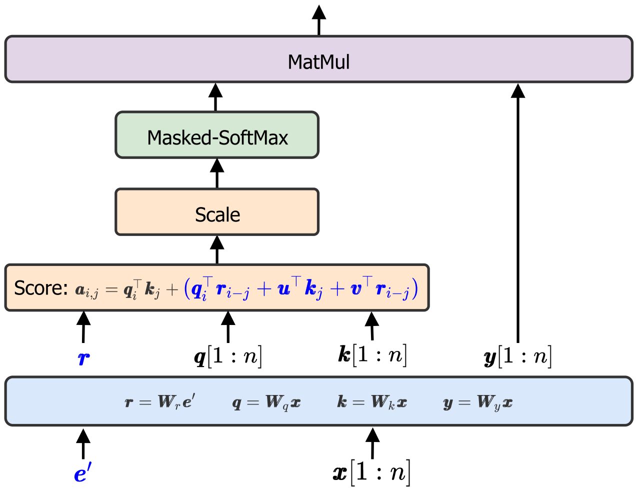

MatMul
Masked-SoftMax
Scale
Score: $\boldsymbol{a}_{i,j} = \boldsymbol{q}_i^\top \boldsymbol{k}_j + (\boldsymbol{q}_i^\top \boldsymbol{r}_{i-j} + \boldsymbol{u}^\top \boldsymbol{k}_j + \boldsymbol{v}^\top \boldsymbol{r}_{i-j})$
$\boldsymbol{r}$
$\boldsymbol{q}[1:n]$
$\boldsymbol{k}[1:n]$
$\boldsymbol{y}[1:n]$
$\boldsymbol{r} = \boldsymbol{W}_r\boldsymbol{e}'$ $\boldsymbol{q} = \boldsymbol{W}_q\boldsymbol{x}$ $\boldsymbol{k} = \boldsymbol{W}_k\boldsymbol{x}$ $\boldsymbol{y} = \boldsymbol{W}_y\boldsymbol{x}$
$\boldsymbol{e}'$
$\boldsymbol{x}[1:n]$

Fig. 4. Illustration of the self-attention part with single head in RPE-Transformer block, where the blue part inside the brackets shows the main difference, $\boldsymbol{e}'$ is the embeddings of all possible relative positions, $\boldsymbol{W}_r$, $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, $\boldsymbol{W}_y$, $\boldsymbol{u}$ and $\boldsymbol{v}$ are trainable parameters.

to the queries $\boldsymbol{q}$, keys $\boldsymbol{k}$ and values $\mathbf{y}$ using three different linear functions, and $\boldsymbol{e}'$ is the PRE vector adapted from Dai *et al.* (2019) [29], which is linearly projected to $\boldsymbol{r}$. The blue bracketed part indicates the extra part of our RPE-Transformer compared to the standard Transformer, where an extra scoring part is used to adjust the attention weights of different relative positions. By this way, we are able to enhance the short-term connections according to the optimization objective of our task.

Based on the updated multi-head self-attentive block, the encoder outputs of the improved Transformer (RPE-Transformer, for short) can be formalized as:

$$\boldsymbol{h}_1^l \cdots \boldsymbol{h}_n^l = \text{RPE-Transformer}(\boldsymbol{h}_1^{l-1} \cdots \boldsymbol{h}_n^{l-1}) \tag{5}$$

where $\boldsymbol{h}_i^0 = \boldsymbol{e}_i$ is the character representations solely, and $\boldsymbol{h}_1^{1:L^{\text{E}}} \cdots \boldsymbol{h}_n^{1:L^{\text{E}}}$ ($l \in [1, L^{\text{E}}]$) are the encoder outputs.

C. Decoding

As mentioned before, we decompose the character-level dependency parsing into two subtasks: (1) unified segmentation and tagging, and (2) dependency parsing, and thus our decoding part is targeted to the two subtasks, respectively. We produce the outputs for both subtasks and then merge them to form a character-level dependency tree. First, we design a subtask-aware aggregation layer over the encoder outputs $\boldsymbol{h}_1^{1:L^{\text{E}}} \cdots \boldsymbol{h}_n^{1:L^{\text{E}}}$ to obtain subtask-dependent features, which can be formalized as follows:

$$\boldsymbol{h}_i^{\text{s\&t}}, \boldsymbol{h}_i^{\text{dp}} = \sum_{l=1}^{L^{\text{E}}} \boldsymbol{\lambda}_l^{\text{s\&t}} \boldsymbol{h}_i^l, \sum_{l=1}^{L^{\text{E}}} \boldsymbol{\lambda}_l^{\text{dp}} \boldsymbol{h}_i^l \tag{6}$$

where $\boldsymbol{\lambda}^{\text{s\&t}}$ is used for the unified segmentation and tagging, and $\boldsymbol{\lambda}^{\text{dp}}$ is for dependency parsing. Both the two parameters are softmax-normalized, with summed values equaling to 1.

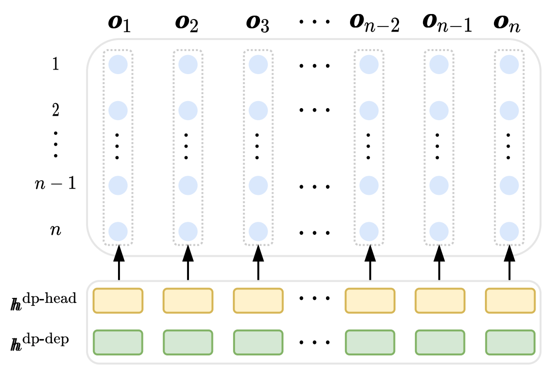


Fig. 5. Illustration of the candidate dependency links between head characters and dependent characters in a sentence.

Unified Segmentation and Tagging. The subtask is essentially a character-level sequence labeling problem. For each character, our model predicts a label with joint word boundary and POS tag information, which can be formulated as a multi-class classification problem. Given the input $\boldsymbol{h}_1^{\text{s\&t}} \cdots \boldsymbol{h}_n^{\text{s\&t}}$, at position i, we score all candidate labels by the following equations:

$$\boldsymbol{o}_i^{\text{s\&t}} = \boldsymbol{W}_{ST}\boldsymbol{h}_i^{\text{s\&t}}, \tag{7}$$

where $\boldsymbol{W}_{ST}$ denotes trainable model parameter. For inference, we utilize the highest-scored label at each position as the final predicted label.

Dependency Parsing. We exploit a biaffine scorer to score each possible dependency link $i \overset{*}{\curvearrowleft} j$ (* is the relation label), following Dozat and Manning (2017) [9]. We first map $\boldsymbol{h}_1^{\text{dp}} \cdots \boldsymbol{h}_n^{\text{dp}}$ into two lower-dimensional feature sets by two parallel MLP layers, which are used to represent it as dependent and as head respectively, and then apply biaffine operations to calculate the score of each candidate dependency link (Fig. 5). The detailed computation is as follows:

$$\begin{aligned} \boldsymbol{h}_i^{\text{dp-dep}}, \boldsymbol{h}_i^{\text{dp-head}} &= \text{MLP}^{\text{dep}}(\boldsymbol{h}_i^{\text{dp}}), \text{MLP}^{\text{head}}(\boldsymbol{h}_i^{\text{dp}}) \\ \boldsymbol{o}_i^{\text{dp-dep}}(\overset{*}{\curvearrowleft} j) &= \text{BIAFFINE}(\boldsymbol{h}_i^{\text{dp-dep}}, \boldsymbol{h}_j^{\text{dp-head}}) \end{aligned} \tag{8}$$

where $*$ indicates the dependency label, and $\curvearrowleft j$ indicates the headed link from a specified position.

Inference. Noticeably, our dependency parsing subtask is performed based on characters, and the difference between the inner- and inter-word dependencies is very evident by the label. Thus, we can actually obtain word segmentation outputs from this subtask by the inner-word dependencies as well. However, this inevitably leads to conflicts with the unified segmentation and tagging during the inference phase, and the inter-word dependencies might be incompatible with the outputs of the unified segmentation and tagging module. In this work, we solve the problem by giving priority to the unified segmentation and tagging outputs. Based on the segmentation and tagging outputs, constrained dependency parsing is performed: inter-word dependency scores are firstly calculated based on only the head characters of the given words, and then the first-order MST algorithm [30] is employed to obtain a valid dependency tree. This problem only exists in the inference phase. There is no conflict between the two subtasks during the training period, as the gold-standard character-level dependency trees are used.

TABLE I
STATISTICS OF THE THREE DATASETS IN OUR EXPERIMENTS

		CTB5	CTB6	CTB7
Training	#sent	18k	23k	31k
	#word	494k	641k	718k
Development	#sent	350	2.1k	10k
	#word	6.8k	60k	237k
Testing	#sent	348	2.8k	10k
	#word	8.0k	82k	245k

D. Training

There are two kinds of losses in our joint model. For unified segmentation and tagging, we use a softmax function over the output score vectors to obtain the probability distribution of all candidate labels, and then compute the cross-entropy objective loss over the gold-standard labels. For dependency parsing, we exploit a similar cross-entropy idea to calculate the loss. Specifically, for each character, we apply softmax operation over all candidate heads and dependency relation labels to obtain their probability distribution, and then calculate the loss for dependency parsing. Finally, we combine the two losses together. The overall process for a single input can be formalized as:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{s\&t}} + \mathcal{L}_{\text{dp}} \\ &= -\sum_i \log \frac{e^{\boldsymbol{o}_i^{\text{s\&t}}[\tilde{t}_i]}}{Z^{\text{s\&t}}} - \sum_i \log \frac{e^{\boldsymbol{o}_i^{\text{dp-dep}}(\overset{*}{\curvearrowleft}\tilde{h}_i)[\tilde{l}_i]}}{Z^{\text{dp}}} \end{aligned} \tag{9}$$

where $Z^{\text{s\&t}} = \sum_t e^{\boldsymbol{o}_i^{\text{s\&t}}[t]}$ and $Z^{\text{dp}} = \sum_{j,l} e^{\boldsymbol{o}_i^{\text{dp-dep}}(\overset{*}{\curvearrowleft} j)[l]}$, which are two normalization factors for probability calculation, $\tilde{t}$ indicates the ground-truth label for unified word segmentation and tagging, and $\tilde{h}$ and $\tilde{l}$ indicate the gold-standard dependency head and label, respectively.

III. EXPERIMENTS

A. Settings

Dataset. We conduct experiments on three benchmark datasets from Chinese Penn Treebank [13], [31], [32], including version 5.0, 6.0 and 7.0, each of which is split into training, development and testing sections according to Zhang *et al.* (2017) [13]. We use CTB5, CTB6, and CTB7 to denote the three datasets for short. Table I shows the statistics of these datasets briefly.

Evaluation. For evaluation, we adopt the word-level F1 score as the major metrics to measure the performance of Chinese word segmentation, POS tagging and dependency parsing, respectively, following Zhang *et al.* (2017) [13]. For POS tagging, we regard a POS tag as correct only when the corresponding word boundaries and tag are both exactly correct. For dependency parsing, we treat a dependency as correct only when the included word pairs are both correctly recognized.[4] We use

[4]Punctuation words are ignored during the evaluation of dependency parsing.

TABLE II
MAIN RESULTS ON THE CTB TEST SETS, ALONG WITH DIFFERENT CHARACTER REPRESENTATIONS AND ENCODING MODULES. +IWD DENOTES THE HUMAN-ANNOTATED INNER-WORD DEPENDENCIES ARE USED, AND THE BEST PERFORMANCE IS MARKED BOLD

Model	CTB5				CTB6				CTB7			
	SEG	POS	UDEP	LDEP	SEG	POS	UDEP	LDEP	SEG	POS	UDEP	LDEP
Embedding												
BiLSTM	98.10	95.01	84.40	82.67	95.86	91.83	80.71	78.04	95.78	91.26	80.32	77.39
Transformer	98.31	95.26	85.29	83.47	96.23	92.58	81.24	78.65	96.19	92.08	80.36	77.56
RPE-Transformer	**98.47**	**95.59**	**86.83**	**85.23**	**96.37**	**92.89**	**82.88**	**80.49**	**96.37**	**92.38**	**81.38**	**78.71**
RPE-Transformer (+IWD)	98.11	95.19	85.82	84.33	96.35	92.81	82.24	79.81	96.19	92.16	81.31	78.55
BERT (frozen)												
BiLSTM	98.70	96.27	91.62	90.39	97.24	94.54	88.31	86.27	97.11	93.69	86.57	84.36
Transformer	98.58	96.14	91.88	90.46	97.32	94.63	88.24	86.24	96.98	94.23	86.76	84.39
RPE-Transformer	**98.75**	**96.60**	**91.94**	**90.59**	**97.38**	94.85	**88.52**	**86.51**	97.24	94.38	**87.12**	**84.84**
RPE-Transformer (-Adapter)	98.63	96.34	91.45	89.81	97.22	94.36	88.13	86.17	97.09	93.85	86.59	84.31
RPE-Transformer (+IWD)	98.62	96.54	91.25	89.70	97.36	**94.89**	88.38	86.34	**97.27**	**94.41**	86.84	84.55
BERT (tuned)												
None	98.64	96.37	**91.81**	**90.42**	97.34	94.71	**88.49**	**86.56**	**97.30**	94.25	**87.07**	**84.82**
None (+IWD)	**98.74**	**96.46**	91.34	89.95	**97.39**	**94.82**	88.38	86.28	97.29	**94.30**	87.03	84.77

UDEP to denote the parsing performance without considering dependency labels, and LDEP to denote the labeled parsing performance.

Pre-trained Embedding Representations. We pre-train 200-dimensional character unigram and bigram embeddings as well as word embeddings on the Chinese Gigaword corpus (LDC2011T13) by using *word2vecf*,[5] where word embeddings are only used in our pipeline models for fair comparisons. For BERT, we utilize the publicly released pre-trained Chinese BERT model directly.[6]

Hyper-parameters. There are several key hyper-parameters for our models. The hyper-parameters in our BiLSTM encoder and biaffine decoder are set according to Dozat and Manning (2017) [9]. The dimension of input character representations is set according to the settings of the external resources. For the exploited Transformer encoders, the layer size, the head number, the model size as well as the inner attention hidden size are 6, 8512 and 1024, respectively. For the model based on BERT (frozen), we set $S = 9$, indicating that the last four BERT layers are adapted for feature extraction, and set the dimension size of the down-projection layer inside the adapter module as $768/4 = 192$.

To train the models, we set the batch size to 32 and use the Adam optimizer with learning rate of 3.5e-4, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon =$ 1e-9. The warmup strategy is adopted following Vaswani *et al.* (2017) [23] when the Transformer structures are required to be optimized anywhere. If BERT parameters require fine-tuning, we use a separated AdamW optimizer for these parameters with the maximum learning rate being 2e-5, following Devlin *et al.* (2019) [18] and Loshchilov and Hutter (2019) [33].

B. Main Results

Table II compares the word segmentation, POS tagging and dependency parsing results of our proposed models on the CTB test sets with different character representations and encoders (as mentioned in Section II). First, we examine the model performance by using character embeddings, aiming to compare different encoders. As shown, the Transformer-based models can bring better performance than the baseline BiLSTM. The averaged LDEP score can be boosted by $(0.80 + 0.61 + 0.17)/3 \approx 0.53$ using the standard Transformer. Our proposed RPE-Transformer can obtain more significant gains over the baseline by $(2.56 + 2.45 + 1.32)/3 \approx 2.11$ on average. Second, we look at the results using BERT (frozen). The tendency is similar to the embedding-based models among the baseline BiLSTM, the standard Transformer and the RPE-Transformer. The gaps are much smaller, leading to very small differences between the baseline BiLSTM and the standard Transformer. The RPE-Transformer still performs the best, with an averaged LDEP improvement of $(0.20 + 0.24 + 0.48)/3 \approx 0.31$ over the BiLSTM.

Further, we compare the character-level parsing models with different character representations. Clearly, in contrast to pre-trained character embeddings, BERT (frozen) representations can improve the performance of character-level parsing dramatically. The averaged gap between embedding representations and BERT (frozen) is $(5.36 + 6.02 + 6.13)/3 \approx 5.84$ when the better encoder based on RPE-Transformer is used. As our BERT (frozen) model incorporates the adapter module for better representations, we also carry out the feature ablation experiments to verify its effectiveness. As shown in Table II, we can see that the performance of BERT (frozen) without the adapter will be dropped significantly, resulting in an averaged decrease of $(0.78 + 0.34 + 0.53)/3 = 0.55$ in LDEP.

We also show the results using BERT with fine-tuning. Under this setting, we directly use the BERT outputs as the final encoder outputs (referred to as None), because we find that no more performance gains can be achieved when additional neural structures are used according to our preliminary experiments, and meanwhile it is very concise. We can observe that the parsing performance of the final BERT (frozen) model is comparable to the model based on BERT with fine-tuning, while BERT with fine-tuning need keep an individual copy of BERT weights for

[5]https://bitbucket.org/yoavgo/word2vecf/
[6]BERT: https://github.com/google-research/bert

TABLE III
COMPARISON OF THE PIPELINE AND SEMI-JOINT MODELS AGAINST THE CHARACTER-LEVEL MODELS ON THE CTB TEST SETS

Model	CTB5				CTB6				CTB7			
	SEG	POS	UDEP	LDEP	SEG	POS	UDEP	LDEP	SEG	POS	UDEP	LDEP
Embedding												
SEG + POS + DEP	98.16	94.98	86.28	84.22	96.19	92.35	82.41	80.19	96.14	92.01	81.09	78.40
SEGPOS+DEP	98.24	95.07	86.54	84.53	96.24	92.36	82.51	80.33	96.18	92.05	81.21	78.52
SEG+POSDEP	98.16	95.34	86.51	84.57	96.19	92.45	82.47	80.21	96.14	92.22	81.19	78.46
Character-Level	**98.47**	**95.59**	**86.83**	**85.23**	**96.37**	**92.89**	**82.88**	**80.49**	**96.37**	**92.38**	**81.38**	**78.71**
BERT (frozen)												
SEG + POS + DEP	98.46	95.99	91.45	89.57	96.68	94.31	88.11	86.23	96.75	93.91	86.54	84.41
SEGPOS+DEP	98.55	96.17	91.59	89.77	97.22	94.53	88.25	86.29	97.01	94.18	86.57	84.36
SEG+POSDEP	98.46	96.13	91.65	89.82	96.68	94.40	88.19	86.28	96.75	94.07	86.50	84.32
Character-Level	**98.75**	**96.60**	**91.94**	**90.59**	**97.38**	**94.85**	**88.52**	**86.51**	**97.24**	**94.38**	**87.12**	**84.84**
BERT (tuned)												
SEG + POS + DEP	98.37	96.16	91.44	89.62	96.87	94.28	88.18	86.21	96.83	94.03	86.68	84.37
SEGPOS+DEP	98.46	96.23	91.52	89.81	97.09	94.56	88.28	86.32	97.07	94.17	86.91	84.56
SEG+POSDEP	98.37	96.19	91.48	89.67	96.87	94.31	88.21	86.29	96.83	94.08	86.79	84.48
Character-Level	**98.64**	**96.37**	**91.81**	**90.42**	**97.34**	**94.71**	**88.49**	**86.56**	**97.30**	**94.25**	**87.07**	**84.82**

this concrete model. In addition, as our method can incorporate the human-annotated inner-word dependencies naturally by the dependency parsing subtask, we also check the influence of them based on the best models of each word representation method (i.e., shown by +IWD). The results demonstrate that slightly decreased performances will be achieved when these inner-word dependencies are used.

Additionally, the performance of word segmentation and POS tagging is mostly consistent with that of dependency parsing, while the differences are much smaller, even sometimes insignificant between the BiLSTM and the standard Transformer. The main reason lies in that the baseline model is already very strong. For example, based on the BERT representations, the averaged gap between the two different encoders is less than 0.1. In summary, all the results demonstrate that the proposed RPE-Transformer and BERT representations are important to all of the three tasks.

C. Compared With the Pipeline Models

Apart from the character-level joint learning strategy, we also investigate the following three kinds of pipeline strategies of Chinese word segmentation (CWS), POS tagging and dependency parsing, making comparisons with our character-level joint models:

- **Pipeline** (CWS $\mapsto$ POS $\mapsto$ DEP): we treat CWS, POS tagging and dependency parsing as independent ones, and train them separately.
- **SEGPOS+DEP** (CWS&POS $\mapsto$ DEP): we train CWS and POS tagging jointly while dependency parsing is trained separately.
- **SEG+POSDEP** (CWS $\mapsto$ POS&DEP): we train CWS separately while POS tagging and dependency parsing components are trained jointly. The segmentation output is used for POS tagging and dependency parsing.

All word-based models also exploit the pre-trained word embeddings to make the pipeline stronger.

For the models with embeddings and BERT (frozen) representations, we exploit the RPE-Transformer as the encoder, since it can bring relatively better performance. The detailed hyper-parameters are the same as our character-level models. As shown in Table III, our character-level dependency parsing models can obtain higher F-scores for word segmentation, POS tagging and dependency parsing. The results demonstrate the advantages of the character-level models for the three tasks. In more detail, we find that the three pipeline models show no significant difference compared with each other while our final joint model is better, indicating the joint learning of word segmentation and dependency parsing could be greatly helpful, which has also been demonstrated in Yan *et al.* (2020) [12].

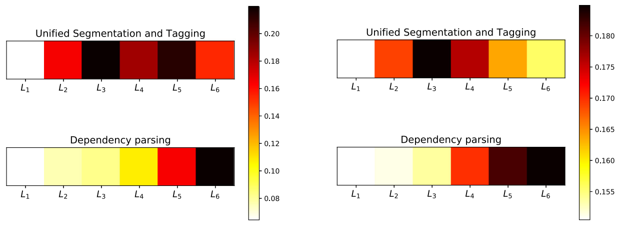
D. Comparisons With Previous Studies

Further, we compare our character-level dependency parsing models with previous closely-related studies, including other joint models of word segmentation, POS tagging and dependency parsing as well [13], [34]–[36]. Table IV shows the comparison results. First, compared with the traditional statistical models, the neural-based joint models can boost parsing performance significantly, except Kurita *et al.* (2017) [37]. By using word or character embeddings, the best UDEP score is advanced from $(82.01 + 76.75 + 75.63)/3 \approx 78.13$ to $(86.83 + 82.88 + 81.38)/3 \approx 83.70$ on average. The averaged score can be further improved with the fine-tuned BERT representations, reaching $(91.81 + 88.49 + 87.07)/3 \approx 89.12$.

We also compare our work to the top-performance models proposed by Yan *et al.* (2020) [12], which is a joint model for Chinese word segmentation and dependency parsing. For fair comparisons, we exclude the POS-tagging part from our models. With embedding inputs, our model is comparable to their joint method. When the BERT representations are integrated, our model can outperform their method significantly, which might be due to the differences in the pre-trained BERT and the decoding strategies. Through the comparisons, we can see that the POS tagging task is helpful to the character-level dependency parsing to some extent, especially the parsing performance. Furthermore, BERT representation can greatly reduce the effectiveness of POS tagging, which is consistent with Zhou *et al.* (2020) [38].

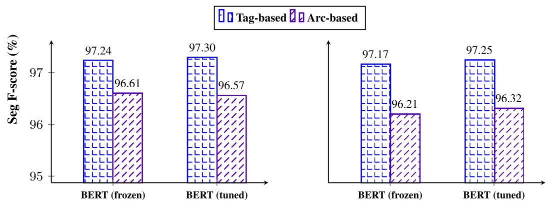
TABLE IV
COMPARISON WITH THE PREVIOUSLY PROPOSED JOINT MODELS AGAINST OURS ON THE CTB TEST SETS

Model	CTB5				CTB6				CTB7			
	SEG	POS	UDEP	LDEP	SEG	POS	UDEP	LDEP	SEG	POS	UDEP	LDEP
Traditional Statistical Models												
Hatori et al. (2012) [34]	97.75	94.33	81.56	–	95.45	91.27	74.88	–	95.42	90.62	73.58	–
Zhang et al. (2014) [13]	97.67	94.28	81.63	–	95.63	91.40	76.75	–	**95.53**	**90.75**	**75.63**	**–**
Yuan et al. (2015) [35]	98.04	94.47	**82.01**	–	–	–	–	–	–	–	–	–
Guo et al. (2016) [36]	**98.31**	**94.84**	81.71	–	–	–	–	–	–	–	–	–
Embeddings												
Kurita et al. (2017) [37]	98.37	94.83	81.42	–	–	–	–	–	95.86	90.91	74.04	–
Yan et al. (2020) [12]	**98.48**	**–**	**87.86**	85.08	–	–	–	–	**96.64**	**–**	**81.80**	77.84
Ours (no POS)	98.21	–	86.71	84.98	95.77	–	82.34	79.84	96.08	–	80.73	77.62
Ours	98.47	**95.59**	86.83	**85.23**	**96.37**	**92.89**	**82.88**	**80.49**	96.37	**92.38**	81.38	**78.71**
BERT												
Yan et al. (2020) [12]	98.46	–	89.59	85.94	–	–	–	–	97.06	–	85.06	80.71
Ours (frozen, no POS)	98.47	–	91.78	90.41	97.25	–	88.46	86.33	97.17	–	87.04	84.61
Ours (frozen)	**98.75**	**96.60**	**91.94**	**90.59**	**97.38**	**94.85**	**88.52**	86.51	97.24	**94.38**	**87.12**	**84.84**
Ours (tuned, no POS)	98.57	–	91.79	90.38	97.32	–	88.44	86.49	97.25	–	86.93	84.68
Ours (tuned)	98.64	96.37	91.81	90.42	97.34	94.71	88.49	**86.56**	**97.30**	94.25	87.07	84.82



(a) Embedding (b) BERT (frozen)

Fig. 6. Visualization of softmax-normalized weights on different self-attentive encoder layers. $L_x (x \geq 1)$ denotes the x-th hidden layer. A darker color indicates a higher weight.



(a) Full Models (b) Without POS tagging

Fig. 7. Word segmentation performance by using sequence labeling (i.e., Tag-based) and inner-word dependency parsing (i.e., Arc-based), respectively.

E. Analysis

In this section, we conduct detailed experimental analyses on the CTB7 test set, aiming for comprehensive understandings of our character-level dependency parsing models.

Word Segmentation by Dependency Parsing. Our proposed model exploits a separated character-tagging scheme for joint word segmentation and POS tagging (i.e., tag-based). In fact, word segmentation can also be achieved by the character-level dependency parsing (i.e., arc-based) as illustrated before, because words can be extracted directly by the inner-word dependencies [12]. Thus, we compare word segmentation performance of the two strategies, respectively. As shown in Fig. 7(a), the arc-based strategy shows slightly worse F-scores than the tag-based strategy of our final model. Further, we remove the influence of POS tagging by performing only word segmentation and dependency parsing, aiming to compare the two strategies fairly. We get the same observation from Fig. 7(b). The results indicate that the unified segmentation and tagging by character-level sequence labeling is preferable.

Visualization Task-Specific Feature Selections. In our character-level models using embeddings and BERT (frozen) as input, we exploit a weighted aggregation layer to capture the feature preference across different tasks (see 6). Here we visualize the learned task-specific weights to see their differences. Fig. 6 shows the visualization results where the best-performing RPE-Transformer is used as the encoder. We can see that the unified segmentation and tagging as well as dependency parsing indeed prefer different layers from the encoder outputs, as the weights are distributed significantly. By deeply examining the weights, we can find that dependency parsing favors the higher layers than word segmentation and POS tagging in both settings, which demonstrates that dependency parsing is at a higher level.

Structural Analysis of Attention. Previously, we have shown that the RPE-Transformer can bring better performance than the standard Transformer when embeddings and BERT (frozen) are used as input. As the major difference may lie in the attention mechanism between the two network structures, hence we show an attention analysis to comprehend the RPE-Transformer in more detail. Fig. 8 shows the comparison results, where one example of normalized attention weights using the same input sentence is visualized in the picture. As shown, we can find that the RPE-Transformer has higher attention weights surrounding the diagonal line, indicating that the local attentions have been greatly enhanced in the RPE-Transformer, which is consistent with our initial design.

Dependency Performance by Arc Distances. We perform fine-grained dependency parsing performance comparisons in terms of arc distance to show the advantages of character-level parsing in contrast to the word-based pipelines. Intuitively, longer distances could be more difficult to be accurately parsed. Here we define the arc distance of a dependency among the head character and dependent character in a sentence (excluding intra-word

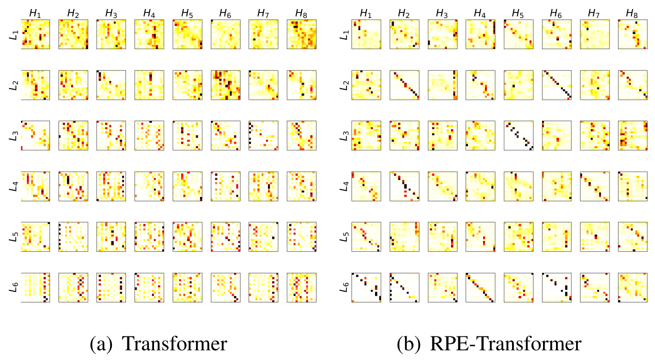


Fig. 8. Illustration of the attention distribution among Transformer and RPE-Transformer encoder by layer / head. A darker color indicates a higher weight.

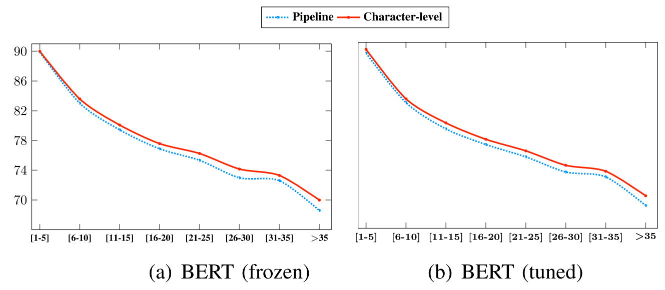


Fig. 9. Dependency F-scores in terms of arc distances.

TABLE V
COARSE-GRAINED POS TYPES

Coarse-Grained POS	Abbr.	Fine-Grained POS
Verb	verb	VV / VC / VE
Noun	noun	NN / NR / NT
Adverb	adv	AD
Pronoun	pron	PN
Preposition	prep	P
Conjunction	conj	CC / CS
Particle	part	DEC / DEG / DER / DEG

arcs), and calculate the F-scores for comparison. Fig. 9 shows the results, where distances are binned by a width of 5. We exhibit the performance of fine-tuned and frozen BERT models. We can observe the overall tendency is matched with our intuition. In addition, the character-level models are better than the pipelines, and the differences are more significant for longer distances.

Dependency Performance by POS Tags. We also study the parsing performance concerning different POS tags. For convenience, we categorize the POS tags into several coarse-grained types, and analyze the performance of several representative types, including nouns, verbs, adverbs, pronouns, prepositions, conjunctions and a subset of particles. Most of the selections are consistent with McDonald and Nivre (2011) [39]. Table V shows the detailed information.

Fig. 10 shows the F-scores of dependency parsing, where both BERT (frozen) and BERT (tuned) are studied to compare the performance of character-level and pipeline models. As a whole, we can see that character-level parsing models outperform the pipelines in both settings for all fine-grained settings, indicating the effectiveness of the character-level architecture. More

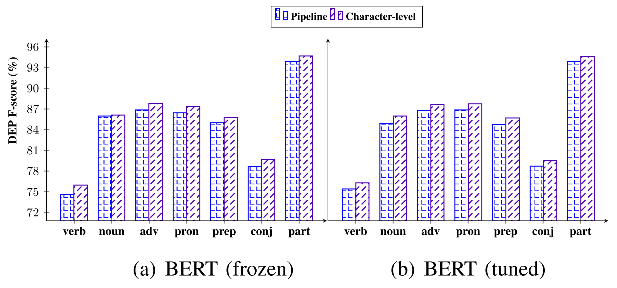


Fig. 10. Dependency performance concerning different POS types.

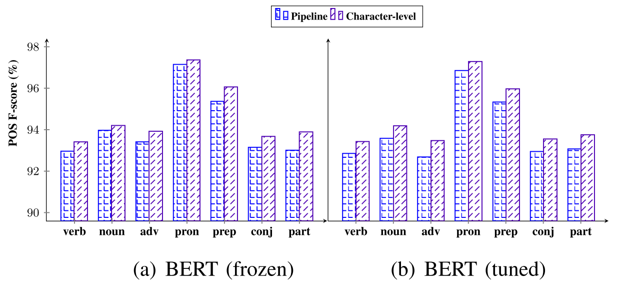


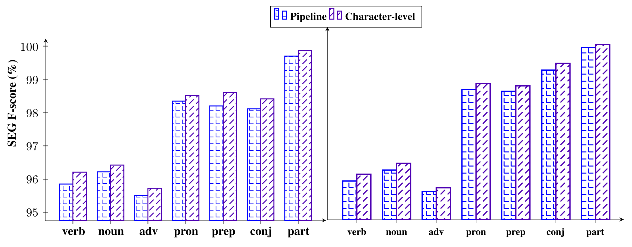
Fig. 11. POS performance of different POS types.

interestingly, we can see that the performance of verb words is obviously lower than others. We examine the dependencies of the verb in more depth, finding that the corresponding dependencies tend to be longer-distance, which could be the main reason accounting for the relatively low performance. Previously, joint models involving POS tagging and dependency parsing can greatly reduce the errors on the particles (as listed in Table V) for Chinese [34]. According to our experimental results, we can see that the gap is no longer larger than other POS types. The major reason may be due to the global feature from deep neural networks, which largely weakens the effectiveness of POS tagging for dependency parsing. The results in Table IV also demonstrate the point, where the performance without POS tagging does not drop as significantly as previous studies based on the traditional statistical models [34].

POS Tagging Performance of Different POS Tags. Here we analyze the POS tagging performance as well. We show the F-scores of our character-level models and their pipelines regarding different POS tags. We also follow Table V to categorize several representative fine-grained POS tags. As shown in Fig. 11, we can see that the character-level models can yield better performance than the pipeline ones. Empirically, the character-level models can not only avoid error propagation from word segmentation, but also use the feedback from dependency parsing, thus they can lead to better performance for fine-grained POS tags. Furthermore, we can also find that the recognition of POS tags such as pron and prep are relatively easier in comparison with other POS tags, since they are closed POS tags, and meanwhile the words of these categories involve smaller ambiguities.[7]

Word Segmentation Performance of Different POS Tags. Finally, we look into the performance of word segmentation. We

[7]https://www.cs.brandeis.edu/ clp/ctb/posguide.3 rd.ch.pdf



(a) BERT (frozen) (b) BERT (tuned)

Fig. 12. Word segmentation performance concerning different POS types.

follow the previous two paragraphs to examine the word-level F-scores in terms of different words, and the same seven types of POS tags are selected for investigation. Fig. 12 shows the comparison results of our character-level joint models and their counterpart pipelines. The results are consistent with the previous comparisons above, where character-level models can consistently give better performance than the pipeline ones. Besides, the word segmentation performance of closed POS tags (i.e., pron, prep, conj, part) is significantly higher than those of open POS tags (i.e., verb, noun, adv), which could be mainly due to that the words of closed POS tags in general have a higher frequency.

IV. Related Work

Dependency parsing has been an active research topic in NLP [1], [3]–[5], [7], [9], [10], [40]. There have been a variety of proposed models, where the representative architectures include the transition-based [6], [41]–[43] and graph-based [1], [7]–[9], [44] approaches. Deep neural models have achieved great success for the tasks [6], [9], [11], [12], [45]. Recently, the graph-based biaffine parser has received great interest, which can achieve excellent parsing performance based on the same word representations [9], [11], [12]. In addition, contextualized word representations such as ELMo and BERT can give further improved performance [11], [46]. Our work focuses on adaptively extending the word-level graph-based biaffine dependency parsing model to the character-level parsing.

Character-level parsing for Chinese has been investigated since very early [13]–[16], [47], [48], which is a natural alternative framework for joint word segmentation, POS tagging and dependency parsing [34]–[36]. Zhang *et al.* (2017) [13] have studied the character-level Chinese dependency parsing comprehensively based on the transition-based framework with traditional statistical models. Under the neural setting, the most representative work includes Li *et al.* [16] and Yan *et al.* (2020) [12]. The latter is closely related to our work, which ignores the POS tagging part. Our work offers the most comprehensive investigations for the graph-based character-level Chinese dependency parsing under the deep neural settings.

V. Conclusion

In this work, we made a comprehensive study on character-level Chinese dependency parsing based on deep graph-based neural models. We extended the word-level deep biaffine parsing model to deal with the character-level parsing, adapting the model suitable for joint word segmentation, POS tagging and dependency parsing via a MTL framework with an additional unified word segmentation and POS tagging task. We investigated different character representations including embedding-based and BERT (frozen or fine-tuning), and also exploited several state-of-the-art neural network structures for advanced feature abstraction including deep BiLSTM, Transformer and (improved) RPE-Transformer.

Experiments were conducted on three widely-used Chinese benchmark datasets, namely CTB5, CTB6 and CTB7, respectively. The results showed that both BERT and our improved RPE-Transformer can obtain better performance compared with the corresponding baselines. Our joint model with BERT (frozen) and RPE-Transformer can achieve comparable performance to the model with the fine-tuning BERT encoder. Since the BERT (frozen) model is more parameter-efficient because the share of BERT parameters can become possible across different NLP models, it might be more favorable in real considerations. We examined our character-level joint models under a range of settings, and compared them with the pipeline models as well as previous works in the literature. Extensive analyses are conducted to understand the advantages of the proposed character-level dependency parsing models.

References

[1] R. McDonald, K. Crammer, and F. Pereira, “Online large-margin training of dependency parsers,” in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics*, 2005, pp. 91–98.
[2] J. Nivre and J. Nilsson, “Pseudo-projective dependency parsing,” in *Proc. Assoc. Comput. Linguistics*, 2005, pp. 99–106.
[3] S. Buchholz and E. Marsi, “CoNLL-X shared task on multilingual dependency parsing,” in *Proc. 10th Conf. Comput. Natural Lang. Learn.*, 2006, pp. 149–164.
[4] J. Nivre *et al.*, “The CoNLL 2007 shared task on dependency parsing,” in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, 2007, pp. 915–932.
[5] S. Kübler, R. McDonald, and J. Nivre, “Dependency parsing,” *Synth. Lectures Hum. Lang. Technol.*, vol. 1, no. 1, pp. 1–127, 2009.
[6] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, “Transition-based dependency parsing with stack long short-term memory,” in *Proc. ACL-IJCNLP*, 2015, pp. 334–343.
[7] E. Kiperwasser and Y. Goldberg, “Simple and accurate dependency parsing using bidirectional LSTM feature representations,” *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 313–327, 2016.
[8] W. Wang and B. Chang, “Graph-based dependency parsing with bidirectional LSTM,” in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 2306–2315.
[9] T. Dozat and C. D. Manning, “Deep biaffine attention for neural dependency parsing,” in *Proc. Int. Conf. Learn. Representations*, 2017.
[10] X. Ma, Z. Hu, J. Liu, N. Peng, G. Neubig, and E. Hovy, “Stack-pointer networks for dependency parsing,” in *Proc. Assoc. Comput. Linguistics*, 2018, pp. 1403–1414.
[11] Y. Li, Z. Li, M. Zhang, R. Wang, S. Li, and L. Si, “Self-attentive biaffine dependency parsing,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 5067–5073.
[12] H. Yan, X. Qiu, and X. Huang, “A graph-based model for joint chinese word segmentation and dependency parsing,” *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 78–92, 2020.
[13] M. Zhang, Y. Zhang, W. Che, and T. Liu, “Character-level chinese dependency parsing,” in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, 2014, pp. 1326–1336.
[14] H. Zhao, “Character-level dependencies in chinese: Usefulness and learning,” *Proc. 12th Conf. Eur. Chapter ACL*, 2009, pp. 879–887.

[15] Z. Li and G. Zhou, "Unified dependency parsing of chinese morphological and syntactic structures," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, 2012, pp. 1445–1454.

[16] H. Li, Z. Zhang, Y. Ju, and H. Zhao, "Neural character-level dependency parsing for chinese," in *Proc. AAAI*, 2018.

[17] M. E. Peters *et al.*, "Deep contextualized word representations." in *Proc. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2018, pp. 2227–2237.

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

[19] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[20] Y. Tian *et al.*, "Joint chinese word segmentation and part-of-speech tagging via two-way attentions of auto-analyzed knowledge," in *Proc. 58th Assoc. Comput. Linguistics*, 2020, pp. 8286–8296.

[21] A. C. Stickland and I. Murray, "BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5986–5995.

[22] N. Houlsby *et al.*, "Parameter-efficient transfer learning for NLP," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.

[23] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[24] M. Zhang, N. Yu, and G. Fu, "A simple and effective neural model for joint word segmentation and POS tagging," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 26, no. 9, pp. 1528–1538, Sep. 2018.

[25] Y. Shao, C. Hardmeier, J. Tiedemann, and J. Nivre, "Character-based joint segmentation and pos tagging for chinese using bidirectional RNN-CRF," in *Proc. Int. Joint Conf. Natural Lang. Process.*, 2017, pp. 173–183.

[26] Y. Hao, L. Dong, F. Wei, and K. Xu, "Visualizing and understanding the effectiveness of BERT," in *Proc. EMNLP-IJCNLP*, 2019.

[27] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?," in *Proc. China Nat. Conf. Chin. Comput. Linguistics.* Berlin, Germany: Springer, 2019, pp. 194–206.

[28] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2018, pp. 464–468.

[29] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. Assoc. Comput. Linguistics*, 2019, pp. 2978–2988.

[30] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič, "Non-projective dependency parsing using spanning tree algorithms," in *Proc. Hum. Lang. Technol. Conf., Conf. Empirical Methods Natural Lang. Process.*, 2005, pp. 523–530.

[31] Y. Zhang and S. Clark, "A fast decoder for joint word segmentation and POS-tagging using a single discriminative model," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2010, pp. 843–852.

[32] Y. Wang, J. Kazama, Y. Tsuruoka, W. Chen, Y. Zhang, and K. Torisawa, "Improving chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data," in *Proc. 5th Int. Joint Conf. Natural Lang. Process.*, 2011, pp. 309–317.

[33] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Representations*, 2019.

[34] J. Hatori, T. Matsuzaki, Y. Miyao, and J. Tsujii, "Incremental joint approach to word segmentation, POS tagging, and dependency parsing in chinese," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics*, 2012, pp. 1045–1053.

[35] Z. Yuan, C. Li, R. Barzilay, and K. Darwish, "Randomized greedy inference for joint segmentation, POS tagging and dependency parsing," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2015, pp. 42–52.

[36] Z. Guo, Y. Zhang, C. Su, J. Xu, and H. Isahara, "Character-level dependency model for joint word segmentation, POS tagging, and dependency parsing in chinese," *IEICE Trans. Inf. Syst.*, vol. E99.D, no. 1, pp. 257–264, 2016.

[37] S. Kurita, D. Kawahara, and S. Kurohashi, "Neural joint model for transition-based chinese syntactic analysis," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1204–1214.

[38] H. Zhou, Y. Zhang, Z. Li, and M. Zhang, "Is POS tagging necessary or even helpful for neural dependency parsing?," in *Proc. Natural Lang. Process. Chin. Comput.*, 2020, pp. 179–191.

[39] R. McDonald and J. Nivre, "Analyzing and integrating dependency parsers," *Comput. Linguistics*, vol. 37, no. 1, pp. 197–230, 2011.

[40] J. Nivre and R. McDonald, "Integrating graph-based and transition-based dependency parsers," in *Proc. Assoc. Comput. Linguistics*, 2008, pp. 950–958.

[41] Y. Zhang and J. Nivre, "Transition-based dependency parsing with rich non-local features," *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics: Hum. Lang. Technol.*, pp. 188–193, 2011.

[42] D. Andor *et al.*, "Globally normalized transition-based neural networks," in *Proc. Assoc. Comput. Linguistics*, 2016, pp. 2442–2452.

[43] J. Nivre, "An efficient algorithm for projective dependency parsing." in *Proc. 8th Int. Conf. Parsing Technol.*, 2003, pp. 149–160.

[44] A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer, and N. A. Smith, "Distilling an ensemble of greedy dependency parsers into one MST parser," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1744–1753.

[45] X. Ma and E. Hovy, "Neural probabilistic model for non-projective MST parsing," in *Proc. Int. Joint Conf. Natural Lang. Process.*, 2017, pp. 59–69.

[46] A. Kulmizev, M. de Lhoneux, J. Gontrum, E. Fano, and J. Nivre, "Deep contextualized word embeddings in transition-based and graph-based dependency parsing–A tale of two parsers revisited," in *Proc. EMNLP-IJCNLP*, 2019, pp. 2755–2768.

[47] X. Luo, "A maximum entropy chinese character-based parser," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2003, pp. 192–199.

[48] M. Zhang, Y. Zhang, W. Che, and T. Liu, "Chinese parsing exploiting characters," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics*, 2013, pp. 1326–1336.