# A Joint Model for Graph-Based Chinese Dependency Parsing

Xingchen Li, Mingtong Liu, Yujie Zhang(✉), Jinan Xu, and Yufeng Chen

School of Computer and Information Technology, Beijing Jiaotong University,
Beijing 100044, China
yjzhang@bjtu.edu.cn

**Abstract.** In Chinese dependency parsing, the joint model of word segmentation, POS tagging and dependency parsing has become the mainstream framework because it can eliminate error propagation and share knowledge, where the transition-based model with feature templates maintains the best performance. Recently, the graph-based joint model [19] on word segmentation and dependency parsing has achieved better performance, demonstrating the advantages of the graph-based models. However, this work can not provide POS information for downstream tasks, and the POS tagging task was proved to be helpful to the dependency parsing according to the research of the transition-based model. Therefore, we propose a graph-based joint model for Chinese word segmentation, POS tagging and dependency parsing. We designed a character-level POS tagging task, and then train it jointly with the model of [19]. We adopt two methods of joint POS tagging task, one is by sharing parameters, the other is by using tag attention mechanism, which enables the three tasks to better share intermediate information and improve each other's performance. The experimental results on the Penn Chinese treebank (CTB5) show that our proposed joint model improved by 0.38% on dependency parsing than the model of [19]. Compared with the best transition-based joint model, our model improved by 0.18%, 0.35% and 5.99% respectively in terms of word segmentation, POS tagging and dependency parsing.

**Keywords:** Dependency parsing · Graph-based · Joint model · Multi-task learning

## 1 Introduction

Chinese word segmentation, part-of-speech (POS) tagging and dependency parsing are three fundamental tasks for Chinese natural language processing, whose accuracy obviously affects downstream tasks such as semantic comprehension, machine translation and question-answering. The traditional method is usually following pipeline way: word segmentation, POS tagging and dependency parsing. However, there are two problems of the pipline way, one is error propagation:

M. Sun et al. (Eds.): CCL 2020, LNAI 12522, pp. 3–16, 2020.
https://doi.org/10.1007/978-3-030-63031-7_1

incorrect word segmentation directly affects POS tagging and dependency parsing, another is information sharing: the tree tasks are strongly related, the label information of one task can help others, but the pipline way cannot exploit the correlations among the three tasks.

Using joint model for Chinese word segmentation, POS tagging and dependency parsing is a solution to these two problems. The previous joint models [7,13,21] mainly adopted a transition-based framework to integrate the three tasks. Based on the standard sequential shift-reduce transitions, they design some extra actions for word segmentation and POS tagging. Although these transition-based models maintained the best performance of word segmentation, POS tagging and dependency parsing, its local decision problem led to the low precision of long-distance dependency parsing, which limited the precision of dependency parsing.

Different from the transition-based framework, the graph-based framework has the ability to make global decisions. Before the advent of neural network, the graph-based framework was rarely applied to the joint model due to its large decoding space to calculate. With the development of neural network technology, the graph-based method for dependency parsing improves rapidly and comes back into researchers' vision. [19] firstly proposed a graph-based unified model for joint Chinese word segmentation and dependency parsing with neural network and attention mechanism, which is superior to the best transition-based joint model in terms of word segmentation and dependency parsing. This work without POS tagging task shows that dependency parsing task is beneficial to Chinese word segmentation.

Chinese word segmentation, POS tagging and dependency parsing are three highly correlated tasks and can improve each other's performance. Dependency parsing is beneficial to word segmentation and POS tagging, while word segmentation and POS tagging are also helpful to dependency parsing, which has been demonstrated by considerable work on the existing transition-based joint model of three tasks. We consider that joint POS tagging task can further improve the performance of dependency parsing. In addition, it makes sense of the model to provide POS information for downstream tasks. For these reasons, this paper proposes a graph-based joint model for word segmentation, POS tagging and dependency parsing. First, we design a character-level POS tagging task, and then combine it with a graph-based joint model for word segmentation and dependency parsing ([19]. As for the joint approach, this paper proposes two ways, one is to combine the two tasks by hard sharing parameters ([3]) and the other is combine the two tasks by introducing tag attention mechanism in the shared parameter layer. Finally, we analyze our proposed models on the Chinese treebank (CTB5) dataset.

## 2 The Proposed Model

In this section, we introduce our proposed graph-based joint model for Chinese word segmentation, POS tagging and dependency parsing. Through the joint

POS tagging task, we explore the joint learning method among multiple tasks and seek for a better joint model to improve the performance of Chinese dependency parsing further.

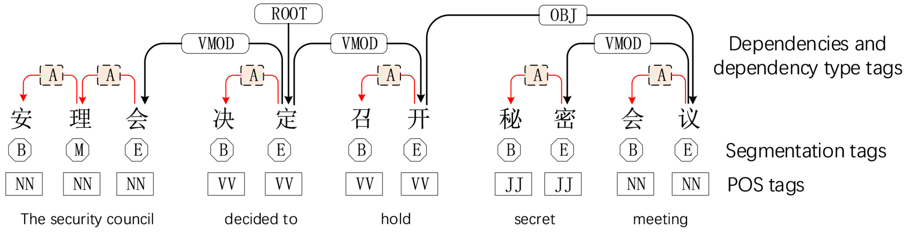


**Fig. 1.** An example of a character-level dependency tree

### 2.1 Character-Level Chinese Word Segmentation and Dependency Parsing

This paper refers to [19]'s approach of combining word segmentation and dependency parsing into a character-level dependency parsing task. Firstly, we transform the word segmentation task to a special arc prediction problem between characters. Specifically, we treat each word as a dependency subtree, and the last character of the word is the root node, and for other characters, the next character is its head node. For example, the root node of the dependency subtree of the word "秘密" is "密", and the head node of the character "秘" is "密", which constitutes an intra-word dependency arc of "秘←密". To distinguish it from the dependencies between words, a special dependency label "Append(A)" was added to represent the dependencies between characters within a word. We use the last character in each word (the root node of the dependency subtree) as a representation of this word, and the dependency between words can be replaced by the dependency between last characters of each word. For example, the dependency relationship "安理会←决定" is transformed into "会←定". Figure 1 shows an example of CTB5 dataset being converted to a character-level dependency tree.

### 2.2 Character-Level POS Tagging

In order to transform the POS tagging into a character-level task, this paper adopts the following rules to convert the POS tag of words into POS tag of each character: the POS tag of each character is the POS tag of the word it is in. In predicting word's POS tag, it is represented by the POS tag of last character of the word. For example, if the predicted POS tag sequence of the word "安理会" is "NN, VV, NN", then the POS tag "NN" of the last character "会" is taken as the POS tag prediction result of the whole word. It is important to note that a word's POS tag is predicted correctly only if the word segmentation is predicted correctly and the last charater's POS tag is also predicted correctly.

### 2.3 Graph-Based Joint Model for Word Segmentation, POS Tagging and Dependency Parsing

According to Sects. 2 and 2.1, after converting three tasks into two character-level tasks, we designed a shared deep Bi-LSTM network to encode the input characters and obtain contextual character vectors. As shown in Fig. 2, given the input sentence (character sequence) $X = \{x_1, ..., x_n\}$. Firstly, vectorize each character $x_i$ to get vector $e_i$, which consists of two parts, one is pre-trained vector $p_i$ which is fixed during training, and the other is randomly initializing embeddings $s_i$ which can be adjusted in training. Element-wise adds the pre-trained and random embeddings as the final input characters' embedding $e_i$, that is $e_i = p_i + s_i$. Then we feed the characters' embedding into multi-layer Bi-LSTM network, and get each character's contextual representation $C = \{c_1, ..., c_n\}$.

$$\overrightarrow{c_i} = \overrightarrow{\text{LSTM}}(e_i, \overrightarrow{c}_{i-1}, \overrightarrow{\theta});\ \overleftarrow{c_i} = \overleftarrow{\text{LSTM}}(e_i, \overleftarrow{c}_{i+1}, \overleftarrow{\theta});\ c_i = \overrightarrow{c_i} \oplus \overleftarrow{c_i} \tag{1}$$

After the contextual character vectors are obtained, the character-level POS tagging and dependency parsing are carried out respectively. We adopted the graph-based framework to analyze the character-level dependency parsing task. By taking each character as a node on the graph, and taking the possibility of forming a dependency relationship between characters as a probability directed edge between nodes (from the head node points to the dependency node), we can define dependency parsing as finding a dependency tree with the highest probability that conforms to the dependency grammar on a directed complete graph. The process of dependency parsing contains two subtasks: prediction of dependency relationship and prediction of dependency relationship type.

**Prediction of Dependency Relationship:** We use $x_i \leftarrow x_j$ to represent the dependency relation between $x_i$ as the dependency node and $x_j$ as the head node. After context encoding, each character obtains a vector representation $c_i$. Considering that each character has the possibility of being a dependency node and a head node, we use two vectors $d_i^{arc}$ and $h_i^{arc}$ to represent them respectively, and get them from $c_i$ through two different MLP, as shown in formula (2).

$$d_i^{arc} = \text{MLP}_d^{arc}(c_i);\ h_i^{arc} = \text{MLP}_h^{arc}(c_i) \tag{2}$$

To calculate the probability $s_{ij}^{arc}$ of $x_i \leftarrow x_j$, we use biaffine attention mechanism proposed by [5].

$$s_{ij}^{arc} = \text{Biaffine}^{arc}(h_j^{arc}, d_i^{arc}) = h_j^{arc} U^{arc} d_i^{arc} + h_j^{arc} u^{arc} \tag{3}$$

where $U^{arc}$ is a matrix whose dimension is $(d_c, d_c)$, and the $d_c$ is the dimension of vector $c_i$, $u^{arc}$ is a bias vector. After we get the scores of all head nodes of the $i$-th character, we select the max score node as its head.

$$s_i^{arc} = [s_{i1}^{arc}, ..., s_{in}^{arc}];\ y_i^{arc} = \arg\max(s_i^{arc}) \tag{4}$$

**Prediction of Dependency Relatoinship Type:** After obtaining the best predicted unlabeled dependency tree, we calculate the label scores $s_{ij}^{label}$ for each

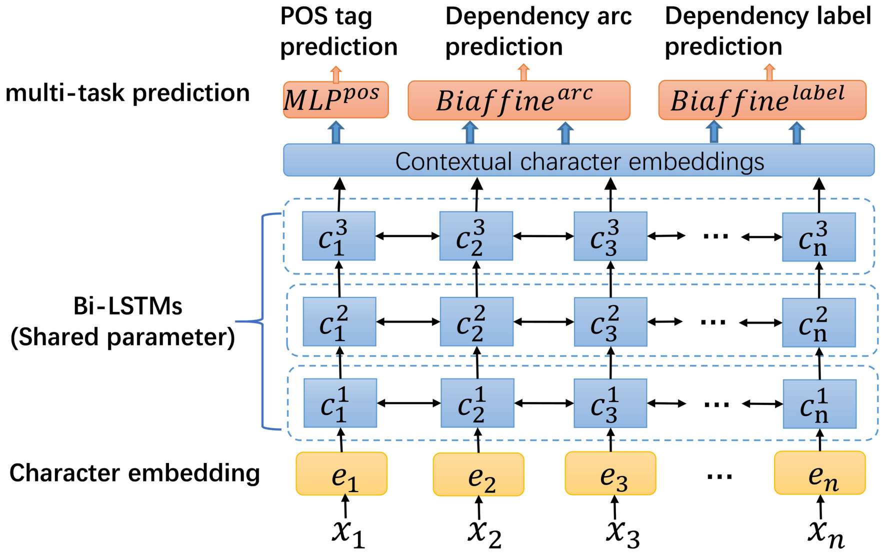

POS tag prediction
Dependency arc prediction
Dependency label prediction
multi-task prediction
$MLP^{pos}$
$Biaffine^{arc}$
$Biaffine^{label}$
Contextual character embeddings
Bi-LSTMs (Shared parameter)
$c_1^3$ $c_2^3$ $c_3^3$ … $c_n^3$
$c_1^2$ $c_2^2$ $c_3^2$ … $c_n^2$
$c_1^1$ $c_2^1$ $c_3^1$ … $c_n^1$
Character embedding
$e_1$ $e_2$ $e_3$ … $e_n$
$x_1$ $x_2$ $x_3$ $x_n$

**Fig. 2.** A joint model of segmentation, POS tagging and dependency parsing with parameter sharing

dependency relationship $x_i \leftarrow x_j$. In our joint model, the arc labels set consists of the standard word-level dependency labels and a special label "A" indicating the intra-dependency within a word. We also use two vectors $d_i^{label}$ and $h_i^{label}$ to represent them respectively, and get them from $c_i$ through two different MLP, and we use another biaffine attention network to calculate the label scores $s_{ij}^{label}$.

$$d_i^{label} = \mathrm{MLP}_d^{label}(c_i);\ \ h_i^{label} = \mathrm{MLP}_h^{label}(c_i) \tag{5}$$

$$s_{ij}^{label} = \mathrm{Biaffine}^{label}(h_j^{label}, d_i^{label}) = h_j^{label} U^{label} d_i^{label} + (h_j^{label} \oplus d_i^{label}) V^{label} + b \tag{6}$$

where $U^{arc}$ is a tensor whose dimension is $(k, d_c, d_c)$, k is the number of dependency relationship labels, and $V^{label}$'s dimension is $(k, 2d_c)$, and $b$ is a bias vector. The best label of the dependency relationship $x_i \leftarrow x_j$ is:

$$y_{ij}^{label} = \arg\max(s_{ij}^{label}) \tag{7}$$

**Prediction of POS Tagging:** We use multi-layer perceptron (MLP) to calculate the probability distribution of the POS tag for each character.

$$s_i^{POS} = \mathrm{MLP}^{POS}(c_i) \tag{8}$$

The best POS tag of the character $x_i$ is

$$y_i^{POS} = \arg\max(s_i^{POS}) \tag{9}$$

**Loss Function for Joint Model:** For the three tasks described above, we adopt cross-entropy loss for all of them, and the results are denoted as $Loss_{arc}, Loss_{dep}, Loss_{pos}$ respectively. The common way to deal with the loss of multiple tasks is to add them together, but this way does not balance the loss of each task. Therefore, we adopt the method proposed by [10], that is using uncertainty to weigh losses for three tasks.

$$\mathcal{L}(\theta) = \frac{1}{\delta_{arc}^2} Loss_{arc} + \frac{1}{\delta_{dep}^2} Loss_{dep} + \frac{1}{\delta_{pos}^2} Loss_{pos} + log\delta_{arc}^2 + log\delta_{dep}^2 + log\delta_{pos}^2 \tag{10}$$

### 2.4 Introduction of Tag Attention Mechanism

The above model joint the three tasks through sharing Bi-LSTM layers to encode the contextual character's information. However, there is no explicit representation of the POS information in the shared encoding layers, the POS tagging task cannot provide the predicted information for word segmentation and dependency parsing. Therefore, we introduce the vector representation of the POS tag and propose the tag attention mechanism (TAM) to integrate the POS information of contextual characters into the vector representation of each character, so that the POS information of the contextual character can also be used in the word segmentation and dependency parsing. This structure is similar to the hierarchically-refined label attention network (LAN) proposed by [4], but we use it to obtain POS information of each layer for subsequent character-level dependency parsing tasks. LAN differs from TAM in that LAN only predicts at the last layer while TAM predicts at each layer. We have tried to predict only at the last layer, but the result of segmentation and dependency parsing is slightly lower than predicting at each layer. The model is shown in Fig. 3.

Firstly, we vectorize the POS tags. Each POS tag is represented by a vector $e_i^t$, and the represents of the set of POS tags denoted as $E^t = \{e_1^t, ..., e_m^t\}$, which is randomly initialized before model training, and then is adjusted during the model training. Then, we calculate the attention weight between the contextual character vectors and POS tag vectors:

$$\alpha = \text{softmax}(\frac{QK^T}{\sqrt{d_c}}) \tag{11}$$

$$E^+ = \text{Attention}(Q, K, V) = \alpha V \tag{12}$$

$$C^+ = \text{LayerNorm}(C + E^+) \tag{13}$$

where Q, K, V are matrices composed of a set of queries, keys and values. We set $Q = C, K = V = E^t$. The $i$-th line of $\alpha$ represents the POS tag probability distribution of the $i$-th character of the sentence. According to this probability distribution $\alpha$, we calculate the representation of predicted POS tag of each character of the sentence, and it is denoted as $E^+$. The $E^+$ is added to the contextual vectors $C$ as the POS tag information. After layer normalization([1], we can obtain the character vectors ($C^+$) containing the POS information, and

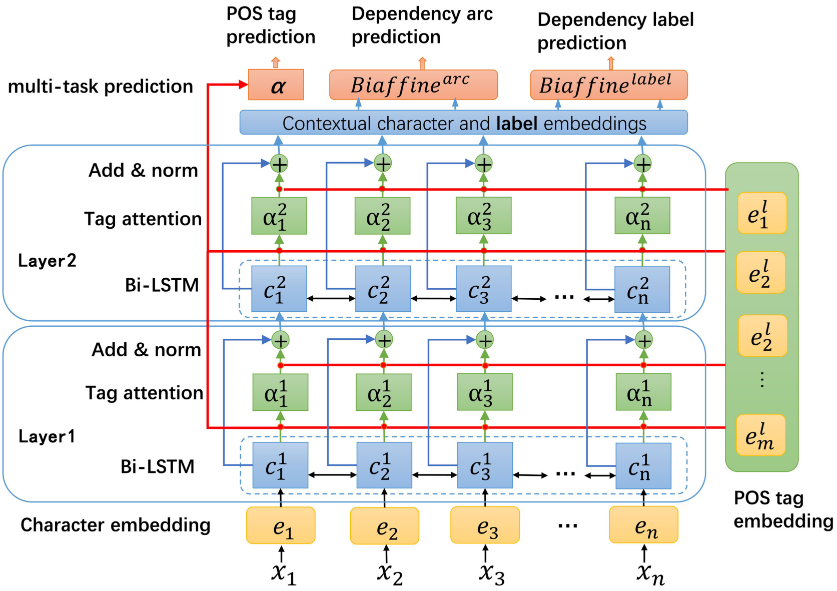


**Fig. 3.** A joint model of segmentation, POS tagging and dependency parsing with tag attention mechanism

then take it as the input of the next Bi-LSTM layer. After the second layer of Bi-LSTM encoding, each character vector we get will contain every characters' POS information, which can be used by word segmentation and dependency parsing.

When the tag attention mechanism is applied, the $i$-th line of the calculated attention weight for each layer is the POS tag distribution of the $i$-th character. Different from the prediction method of POS tagging in previous model, we added the attention weights of all layers as the final POS tag distribution:

$$s_i^{POS} = \sum_{j}^{m} \alpha_i^j \tag{14}$$

where, m is the number of layers. The prediction of POS tag is:

$$y_i^{POS} = \arg\max(s_i^{POS}) \tag{15}$$

For word segmentation and dependency parsing, we use the same approach as the previous model. For the losses of three tasks, we also use the same way to calculate it as the previous model.

# 3 Experiment

## 3.1 Dataset and Evaluation Metrics

We conducted experiments on the Penn Chinese Treebank5 (CTB-5). We adopt the data splitting method as same as previous works [7,13,19]. The training set is from section 1∼270, 400∼931 and 1001∼1151, the development set is from section 301∼325, and the test set is from section 271∼300. The statistical information of the data is shown in Table 1.

**Table 1.** The statistics of the dataset.

| Dataset | Sentence | Word | Character |
|---|---|---|---|
| Training | 16k | 494k | 687k |
| Develop | 352 | 6.8K | 31k |
| Test | 348 | 8.0k | 81k |

Following previous works [9,13,19], we use standard measures of word-level F1 score to evaluate word segmentation, POS tagging and dependency parsing. F1 score is calculated according to the precision P and the recall R as $F = 2PR/(P + R)$ [9]. Dependency parsing task is evaluated with the unlabeled attachment scores excluding punctuations. The output of POS tags and dependency arcs cannot be correct unless the corresponding words are correctly segmented.

## 3.2 Model Configuration

We use the same Tencent's pre-trained embeddings [17] and configuration as [19], and the dimension of character vectors is 200. The dimension of POS tag vectors is also 200. We use with 400 units for each Bi-LSTM layer and the layer numbers is 3. Dependency arc MLP output size is 500 and the label MLP output size is 100. The dropout rates are all 0.33.

The models are trained with Adam algorithm [11] to minimize the total loss of the cross-entropy of arc predictions, label predictions and POS tag predictions, which using uncertainty weights to combine losses. The initial learning rate is 0.002 annealed by multiplying a fix decay rate 0.75 when parsing performance stops increasing on development sets. To reduce the effects of "gradient exploding", we use gradient clip of 5.0 [16]. All models are trained for 100 epochs.

## 3.3 Results

We conduct comparison of our models with other joint parsing models. The model shown in Fig. 2 is denoted as Ours and the model shown in Fig. 3 as Ours-TAM (with tag attention mechanism). The comparison models include three types: one is the transition-based joint models with feature templates [7,13,21], the other is the transition-based joint models with neural network [13](4-g, 8-g), and the third is the graph-based model with neural network without POS tagging task [19]. The results are shown in Table 2[1].

**Table 2.** Performance comparison of Chinese dependency parsing joint models.

| Model | Framework | SEG | POS | DEP |
|---|---|---|---|---|
| Hatori12 [7] | Transition | 97.75 | 94.33 | 81.56 |
| Zhang14 [21] | Transition | 97.67 | 94.28 | **81.63** |
| Kurita17 [13] | Transition | **98.24** | **94.49** | 80.15 |
| Kurita17(4-g) [13] | Transition | 97.72 | 93.12 | 79.03 |
| Kurita17(8-g) [13] | Transition | 97.70 | **93.37** | 79.38 |
| Yan19 [19][1] | Graph | **98.47** | — | **87.24** |
| Ours | Graph | 98.34 | 94.60 | **87.91** |
| Ours-TAM | Graph | **98.42** | **94.84** | 87.62 |

From the table, we see that transition-based joint models using feature templates maintain the best performance in word segmentation, POS tagging and dependency parsing for a long time. Although [13](4-g, 8-g) adopted the neural network approach, it still didn't surpass the joint model with feature templates. While, the graph-based joint model [19] obtained the better performance in word segmentation and dependency parsing than all transition-based model.

Our models Ours and Ours-TAM exceeded [19] 0.67 and 0.38% points respectively in dependency parsing, indicating that the POS tag information contributes to dependency parsing. Although they are 0.13 and 0.05% points lower than [19] on word segmentation task respectively, they still exceed the best transition-based joint model with feature templates [13]. [19] does not have POS tagging task, but our models have, and its performance exceeded that of the previous best joint model [13] by 0.11 and 0.35% points respectively, indicating that after the introduction of POS tagging, other tasks such as dependency parsing are also helpful for POS tagging task itself.

### 3.4 Detailed Analysis

We will further investigate the reasons for the improvement of dependency parsing after the combination of POS tagging task. For a dependency relationship $x_i \leftarrow x_j$, we use $X \leftarrow Y$ to represent its POS dependency pattern, the $X$ is the POS tag of $x_i$, and the $Y$ is the POS tag of $x_j$. We calculated the distribution of $Y$ for each $X$ in training set and found that the probability between some $X$ and $Y$

[1] Yan et al. later submitted an improved version Yan20 [20], and the results of word segmentation and dependency parsing reached 98.48 and 87.86, respectively.

was very high. For example, when $X$ was P(preposition), the distribution of $Y$ was {VV(78.5%), DEG(5.1%), ..., NN(3.1%), ... }. In order to verify whether our models can use these POS informations in training dataset, we calculated the accuracy of each POS dependency patterns in test dataset on our models and the re-implemented model of [19]. The patterns on which the accuracy of our models are better than [19] are shown in left part of Fig. 4.

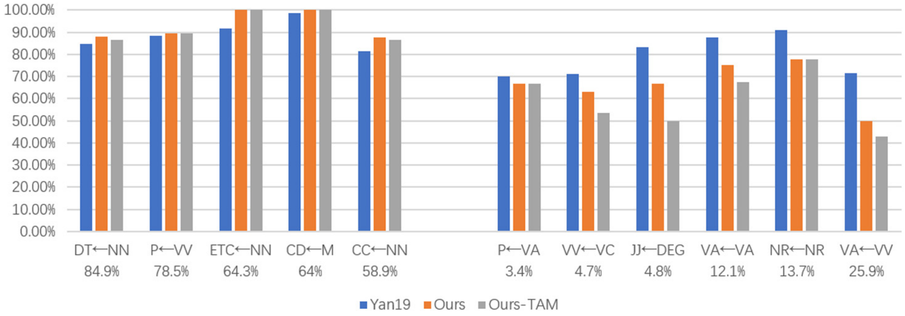


**Fig. 4.** Comparison of precision on different POS tag patterns before and after joint POS tagging task

**Table 3.** Head POS distribution

| Node POS | Head POS distribution | | | | | |
|---|---|---|---|---|---|---|
| DT | NN 84.9% | VV 7.5% | DEG 1.8% | P 1.3% | M 1% | NR 0.8% |
| P | VV 78.5% | DEG 5.1% | VA 3.4% | VE 3.3% | VC 3.1% | NN 3.1% |
| ETC | NN 64.3% | NR 22.5% | VV 10.4% | VA 1.6% | VE 0.2% | VC 0.2% |
| CD | M 64% | NN 20.6% | VV 6.7% | CD 2.7% | DT 1.6% | DEG 1.2% |
| CC | NN 58.9% | VV 20.5% | NR 7.9% | NT 2.3% | VA 2.1% | M 1.9% |

The $X$ of these 5 patterns are {DT, P, ETC, CD, CC}, and the $Y$'s distributions of each $X$ are shown in the Table 3. It is found that all 5 patterns select $Y$ with the highest probability, indicating that our model can fully utilize the POS informations to improve the accuracy of dependencies with these POS dependency patterns. As the example shown in the Fig. 5, when predicting the head node of "从", [19] predicted wrong node "工作", while our models both predicted right node "撤退". The POS tag of "从" is P and the POS tag of correct head node "撤退" is VV whose probability is 78.5%, while the wrong head node 工作 s POS tag is NN whose probability is only 3.1%. Because our models can use these POS informations to exclude the candidate head nodes of low probability POS, thus improving the performance of dependency parsing.

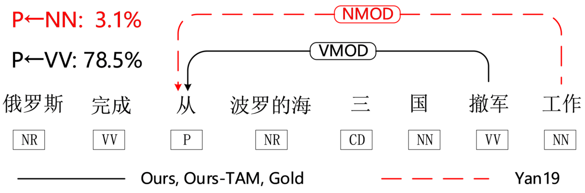
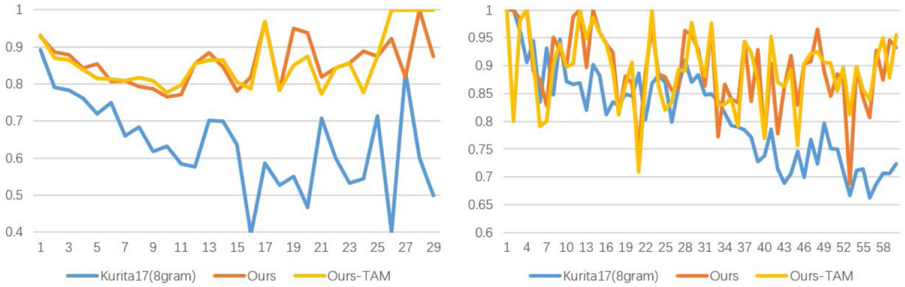


**Fig. 5.** An example of POS information contributes to dependency parsing

Although Ours-TAM achieved better results in segmentation and POS tagging, the dependency parsing was reduced compared with Ours. The right part of the Fig. 4 shows the patterns on which the accuracy of our models are worse than [19]. It can be found that the dependency probability of these patterns is small, and the addition of POS information actually reduces the accuracy. Therefore, Ours-TAM has better POS information, so the accuracy of these patterns is lower than Ours, thus the overall precision of dependency parsing of Ours-TAM decreases compared with that of Ours.



(a) Dependency length

(b) Sentence length

**Fig. 6.** The influence of dependency length and sentence length on dependency parsing

Next, we will investigate the difference between the graph-based joint model and the transition-based joint model in dependency parsing. We compare our graph-based joint models to the transition-based joint model [13] according to dependency length and sentence length respectively. The results are shown in Fig. 6. From the figure, we can see that our proposed joint models on long-distance dependencies have obvious advantages, and the accuracy of the dependency parsing is relatively stable with the increase of sentence length, while the transition-base joint model has an obvious downward trend, which indicates that our graph-based joint model can predict the long-distance dependencies more effectively than transition-based joint model.

## 4 Related Work

[7] proposed a character-level dependency parsing for the first time, which combines word segmentation, POS tagging and dependency parsing, They combined the key feature templates on the basis of the previous feature engineering research on the three tasks, and realized the synchronous processing of the three tasks. [21] annotated the internal structure of words, and regarded the word segmentation task as dependency parsing within characters to jointly process with three tasks. [13] firstly applied neural network to the charater-level dependency parsing. Although these transition-based joint models achieved best accuracy in dependency parsing, they still suffer from the limitation of local decision.

With the development of neural network, the graph-based dependency parsing models [5,12] using neural networks have developed rapidly. these model fully exploit the ability of the bidirectional long short-term memory network (BiLSTM) [8] and attention mechanism [2,18] to capture the interactions of words in a sentence. Different from transition-based models, the graph-based model can make global decision when predicting dependency arcs, but few joint model adopted this framework. [19] firstly proposed a joint model adopting graph-based framework with neural network for Chinese word segmentation and dependency parsing, but they does not use POS tag.

According to the research of existing transition-based joint model, the word segmentation, POS tagging and dependency parsing are three highly correlated tasks that influence each other. Therefore, we consider that integrating POS tagging task into graph-based joint model [19] to further improve the performance and to provide POS information for downstream tasks. We transform the POS tagging task into a character-level sequence labeling task and then combine it and [19] by using multi-task learning. There are many multi-task learning approaches such as [3,14,15] and [6], we use parameter sharing [3] to realize the joint model, and then improve it with tag attention mechanism. Finally, we analyze the models on the CTB5 dataset.

## 5 Conclusion

This paper proposed the graph-based joint model for Chinese word segmentation, POS tagging and dependency parsing. The word segmentation and dependency parsing are transformed into a character-level dependency parsing task, and the POS tagging task is transformed into a character-level sequence labeling task, and we use two ways to joint them into a multi-task model. Experiments on CTB5 dataset show that the combination of POS tagging task is beneficial to dependency parsing, and using the POS tag attention mechanism can exploit more POS information of contextual characters, which is beneficial to POS tagging and dependency parsing, and our graph-based joint model outperforms the existing best transition-based joint model in all of these three tasks. In the future, we will explore other joint approaches to make three tasks more mutually reinforcing and further improve the performance of three tasks.

## References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
3. Baxter, J.: A Bayesian/information theoretic model of learning to learn via multiple task sampling. Mach. Learn. **28**(1), 7–39 (1997)
4. Cui, L., Zhang, Y.: Hierarchically-refined label attention network for sequence labeling. arXiv preprint arXiv:1908.08676 (2019)
5. Dozat, T., Manning, C.D.: Deep biaffine attention for neural dependency parsing. arXiv preprint arXiv:1611.01734 (2016)
6. Hashimoto, K., Xiong, C., Tsuruoka, Y., Socher, R.: A joint many-task model: growing a neural network for multiple nlp tasks. arXiv preprint arXiv:1611.01587 (2016)
7. Hatori, J., Matsuzaki, T., Miyao, Y., Tsujii, J.: 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, vol. 1, pp. 1045–1053. Association for Computational Linguistics (2012)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
9. Jiang, W., Huang, L., Liu, Q., Lü, Y.: A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In: Proceedings of ACL-08: HLT, pp. 897–904 (2008)
10. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7482–7491 (2018)
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Kiperwasser, E., Goldberg, Y.: Simple and accurate dependency parsing using bidirectional LSTM feature representations. Trans. Assoc. Comput. Linguist. **4**, 313–327 (2016)
13. Kurita, S., Kawahara, D., Kurohashi, S.: 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), pp. 1204–1214 (2017)
14. Long, M., Wang, J.: Learning multiple tasks with deep relationship networks. arXiv preprint arXiv:1506.02117 2, 1 (2015)
15. Misra, I., Shrivastava, A., Gupta, A., Hebert, M.: Cross-stitch networks for multi-task learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3994–4003 (2016)
16. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp. 1310–1318 (2013)
17. Song, Y., Shi, S., Li, J., Zhang, H.: 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), pp. 175–180 (2018)
18. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)

19. Yan, H., Qiu, X., Huang, X.: A unified model for joint Chinese word segmentation and dependency parsing. arXiv preprint arXiv:1904.04697 (2019)
20. Yan, H., Qiu, X., Huang, X.: A graph-based model for joint Chinese word segmentation and dependency parsing. Trans. Assoc. Comput. Linguist. **8**, 78–92 (2020)
21. Zhang, M., Zhang, Y., Che, W., Liu, T.: Character-level Chinese dependency parsing. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1326–1336 (2014)