A Top-down Model for Character-level Chinese Dependency Parsing

Yuanmeng Chen, Hang Liu, Yujie Zhang, Jinan Xu and Yufeng Chen

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

Abstract. This paper proposes a novel transition-based algorithm for character-level Chinese dependency parsing that straightforwardly models the dependency tree in a top-down manner. Based on the stack-pointer parser, we joint Chinese word segmentation, part-of-speech tagging, and dependency parsing in a new way. We recursively build the character-based dependency tree from root to leaf in a depth-first fashion, by searching for candidate dependents through the sentence and predicting relation type at each step. We introduce intra-word dependencies into the relation types for word segmentation, and the inter-word dependencies with POS tags for part-of-speech tagging. Since the top-down model provides a global view of an input sentences, the information of the whole sentence and all previously generated arcs are available for action decisions, and all characters of the sentence are considered as candidate dependencies. Experimental results on the Penn Chinese Treebank (CTB) show that the proposed model outperformed existing neural joint parsers by 0.81% on dependency parsing, and achieved the F1-scores of 95.97%, 91.72%, 80.25% for Chinese word segmentation, part-of-speech tagging, and dependency parsing.

Keywords: dependency parsing, Chinese, joint model, pointer networks, stack.

1 Introduction

Dependency parsing is a fundamental technique for natural language processing, whose accuracy obviously affects downstream tasks such as machine translation, question answering, text generation, and so on[1-3]. Word segmentation is needed before Chinese dependency parsing due to lack of obvious delimiters between words. For this problem, the joint models for word segmentation, part-of-speech tagging, and dependency parsing have been proposed by researches [4-7]. Compared with the pipeline models, the joint models integrate the three tasks into one framework to solve the error propagation problem between three tasks, and to be able to use multiple levels (character, word, and phrase level) information for each task. The character-level dependency parsers generally use the classical transition-based framework and improve performance significantly.

However, the weakness of the classical transition-based framework is the lack of a global view of the input sentence. In greedy or beam search, only local information is available for action decisions, and two top nodes of the stack and the first token in the buffer are considered as analysis objects at each step. This restriction usually results in error propagation, particularly in long dependencies that require a larger number of transitions to be built [8].

In order to address this problem, we propose a novel top-down joint model based on stack-pointer networks [9]. Stack-pointer networks realize parsing in a top-down manner, which uses pointer networks [10] to directly find dependencies in sentence for a given word, and predicts the relation types with a multiclass classifier. On the basis of stack-pointer networks, we construct a character-level Chinese dependency framework by designing the intra-word dependencies and predicting of dependency types expanded with POS tags, to integrate Chinese word segmentation, part-of-speech tagging and dependency parsing. With the global view of input sentence, this architecture can capture information from the whole sentence, and make it possible to directly generate arcs between any characters.

We evaluate our model on the Penn Chinese Treebank (CTB version 5.0) and compare with other joint architectures. Comparison results show that our model surpasses the best results of the neural joint models on dependency parsing task. Our contributions are summarized as follows: (1) we propose a top-down algorithm for character-level Chinese dependency parsing, (2) we integrate word segmentation and part-of-speech tagging tasks into a top-down parsing framework, (3) our model surpasses the best neural joint model for dependency parsing.

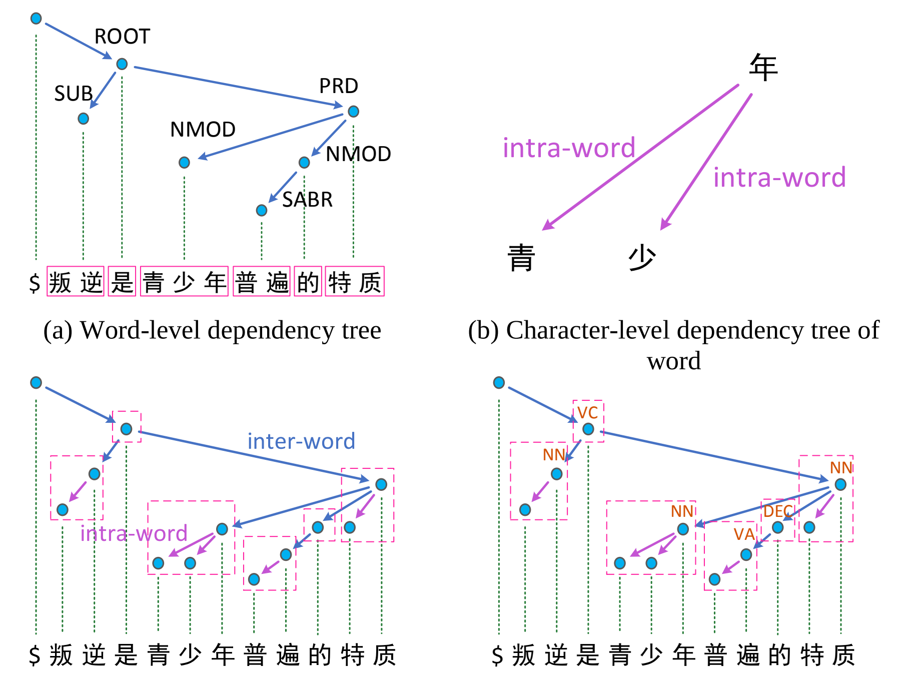
2 Word Segmentation and POS Tagging as Top-down Parsing

2.1 Stack-Pointer Networks

Stack-Pointer Networks (StackPTR) [9] is a dependency framework which has a global view of the input sentence during parsing and a lower computation complexity compared with graph-based parsers. The framework builds dependency arcs in a tree structure without the restriction of left-to-right in classical transition-based parsers, and therefore can consider all words of the sentence as candidates.

For an input sentence $\mathbf{x}=\{w_1, \cdots, w_n\}$, the outputs of StackPTR is $\mathbf{y}=\{p_1, \cdots, p_i, \cdots, p_k\}$, where $p_i=\$, w_{\mathrm{i,1}}, w_{\mathrm{i,2}}, \cdots, w_{\mathrm{i,li}}$ represents the path from the dummy root "\$" to the leaf $w_{\mathrm{i,li}}$, and the two adjacent words $w_{\mathrm{i,j}}$, $w_{\mathrm{i,j+1}}$ on the path is a head-child pair, representing a dependency arc with its type. Fig. 1(a) shows the construction of the dependency tree for the sentence "叛逆是青少年普遍的特质 (Rebellion is a common characteristic of teenagers.)".

StackPTR combines pointer networks with an internal stack. The stack tracks the status of the top-down search and the pointer networks select one child for the word at the top of the stack at each step. Given a stack-top word w_t, the pointer network returns the position p in the input sentence, and then the following actions are conducted according to position p [11]:


ROOT
SUB
PRD
NMOD
NMOD
SABR
$ 叛 逆 是 青 少 年 普 遍 的 特 质
年
intra-word
intra-word
青
少
inter-word
intra-word
$ 叛 逆 是 青 少 年 普 遍 的 特 质
VC
NN
NN
NN
DEC
VA
$ 叛 逆 是 青 少 年 普 遍 的 特 质

(a) Word-level dependency tree

(b) Character-level dependency tree of word

(c) Character-level dependency tree of sentence

(d) Top-down joint dependency parsing

Fig. 1. Illustration of top-down character-level Chinese dependency parsing

- Shift-Attach-p: if $p \neq t$, then the pointed word w_p is considered as a child of w_t, then the parser moves w_p into the stack and build an arc $w_t \rightarrow w_p$, and the relation type will be predicted;
- Reduce: if $p = t$, then w_t is considered to have found all its children, so the parser pops w_t out from the stack.

2.2 Word Segmentation

Since traditional word segmentation makes decisions from left to right as sequence labeling [12-13], it is difficult to directly integrate it into the top-down parsing framework. In order to address this problem, we define an intra-word dependency structure for a Chinese word that consists of a few characters. We take the last character of the word as head (because we know that the segmentation of Chinese words usually performs slightly better in the reverse direction), and each other character as its child. As shown in Fig. 1(b), for the word "青少年(teenagers)", two intra-word arcs "青$\leftarrow$年" and "少$\leftarrow$年" are constructed. In this way, we transform a word to a character-level dependency tree and incorporate such dependency parsing into the top-down framework to implement word segmentation.

As shown in Fig. 1(c), we utilize the prediction mechanism of relation types in StackPTR to predict “intra-word” and “inter-word”. The “intra-word” is used to indicate character-level arc for word segmentation and the “inter-word” is used to indicate word-level arc for dependency parsing. A head and all its children with “intra-word” relation type make up a word.

2.3 POS Tagging

In the prediction mechanism of relation types, we further introduce POS tagging task. We tag each word with POS (as shown in Fig. 1(d)) by adding POS tag to its head character. When the relation type of a dependency arc is predicted as “inter-word”, the child of the arc, which is a head of some word, will be assigned with a POS tag. For simplicity, we predict the relation types and the POS tags simultaneously, so the prediction mechanism can be implemented for both relation types and POS tags by using one classifier.

3 Chinese Joint Dependency Parsing

3.1 Overview

We constructed a top-down joint dependency parsing model based on stack-pointer networks, by integrating word segmentation and POS tagging described in Sec. 2. As shown in Fig. 2, our model consists of three parts: encoder, decoder, and decision layer. The encoder calculates the representation for each character of the input sentence. The decoder regards the top character of the stack S as the head and get its representation at each step. And the decision layer contains a Biaffine attention and a Biaffine classifier. We calculate the correlation scores of the head and all candidates by the attention mechanism and select one with the highest score as child. Then Shift-Attach-p or Reduce is executed according to the position of the selected child with the highest score in the sentence. And the classifier predicts the relation type according to the feature vectors of the head and the child.

We use tag “IN” and “OUT” to indicate “intra-word” and “inter-word” relation types, respectively, and add the POS tag to the “OUT” tag. Specially, we use a meaningless tag "PAD" for Reduce action because there is no arc formed.

3.2 Encoder

The encoder of our model is a deep bidirectional LSTM (Bi-LSTM) [14]. For a given sequence of characters x={c_1, ⋯, c_n}, we lookup embedding vector e_i from the pre-trained embedding matrix for each character c_i, and then feed them into the deep BiLSTM to obtain representations containing context information.

The Bi-LSTM learns the representation {s_1, ⋯, s_n} of every character from two directions respectively as follows:

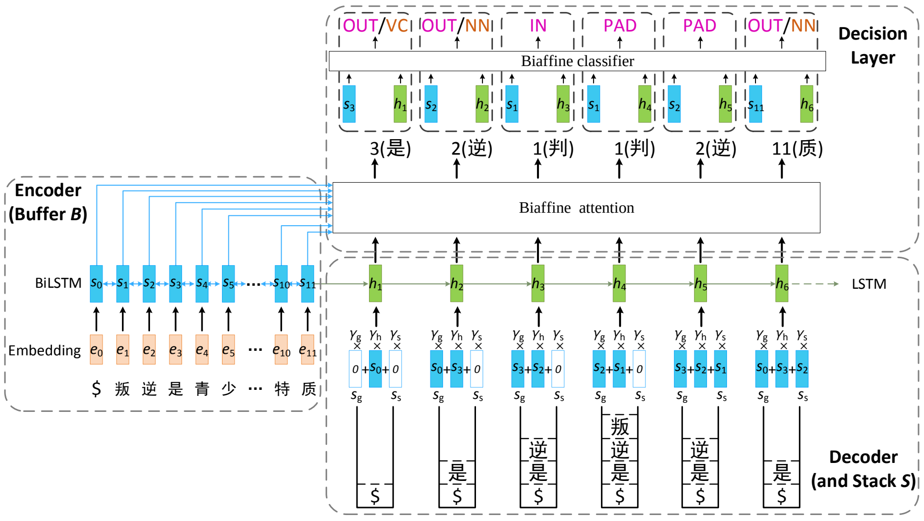


Fig. 2. Joint Chinese dependency parsing model based on StackPTR

$$\overrightarrow{s_i} = \overrightarrow{\mathrm{LSTM}}(\overrightarrow{s_{i-1}}, e_i) \tag{1}$$

$$\overleftarrow{s_i} = \overleftarrow{\mathrm{LSTM}}(\overleftarrow{s_{i+1}}, e_i) \tag{2}$$

$$s_i = \overrightarrow{s_i} \oplus \overleftarrow{s_i} \tag{3}$$

where $\overrightarrow{s_i}$ and $\overleftarrow{s_i}$ represent the hidden states of the forward and the backward LSTM, respectively. $\oplus$ represents the vector concatenating operation. We use the hidden state of the last layer as a representation of each character.

3.3 Decoder

The decoder consists of a uni-directional LSTM. At each time, the vector s_h of the stack-top character looked as a head is fed into the LSTM, and then the hidden state h_t is obtained. Considering that LSTM's hidden state can be passed over time, previously generated information can be implicitly used in every step of decisions.

Higher-order Information. We also use the higher-order information to add to the head as the input of LSTM like StackPTR [9]: grandfather and sibling. Furthermore, we introduce three weights γ_h , γ_g and γ_s for the head, grandfather and sibling, respectively, to control the utilization of the three kinds of information.

3.4 Biaffine Attention and Biaffine Classifier

At each step, all the represents $s_j (s_j \in \{s1, \ldots, sn\})$ of children and the h_t of the head are used in attention mechanism to predict the position p for the action decisions, and the classifier use the h_t and s_p to predict the relation type.

There are different types for attention functions such as dot-product, concatenation (or nonlinear transformation), general (bilinear transformation) and bi-affine [15-16]. We adopt the biaffine attention mechanism following Ma et al. [9]. The calculation for bi-affine attention score function is as follows:

$$\alpha_i^t = h_t^{\mathrm{T}} \mathbf{W} s_i + \mathbf{U}^{\mathrm{T}} h_t + \mathbf{V}^{\mathrm{T}} s_i + b \tag{4}$$

where **W**, **U**, **V** and *b* are parameters, denoting the weight matrix of the bilinear term, the two weight vectors of the linear terms, and the bias, respectively; *h* and *s* represent vector representations of head and child, respectively. It can be seen that the bi-affine function combines the bilinear and the linear transformation, and therefore the advantages of both are brought about. The first three terms correspond to the relationship between h_t and s_i, the information of h_t, and the information of s_i, Therefore, the prediction can be made from a more comprehensive perspective [17].

The Eq (4) is also used in the Biaffine classifier to accept the two inputs of the head and its child. Since the different sets of parameters are set up for every type, the scores of every type are available.

3.5 Training

Children Order. When a head character has multiple children, it is possible that there is more than one valid selection at each step. In order to define a deterministic decoding process to make sure there is only one ground-truth choice at each step, a predefined order for selecting children for this head needs to be introduced. The predefined order of children can have different alternatives, such as *left-to-right* or *inside-out*. In this paper, we adopt them both and the *inside-out* performs better.

LOSS. Given the input sentence $\mathbf{x}=\{c_1, \cdots, c_n\}$, the top-down parsing model is to search for a dependency tree $\mathbf{y}=\{p_1, \cdots, p_k\}$ with a highest probability. We train the model by minimizing the cross-entropy loss as follows:

$$\begin{aligned} L(\theta) &= -\sum_{i=1}^{k} \log P_\theta(p_i | p_{<i}, \mathbf{x}) \\ &= -\sum_{i=1}^{k} \sum_{j=1}^{l_i} \log P_\theta(c_{i,j} | c_{i,<j}, p_{<i}, \mathbf{x}) \end{aligned} \tag{5}$$

where θ represents all parameters of the model. $p_{<i}$ represents paths that have been built completely, k is the total number of paths, l_i represents the total number of characters in p_i, $c_{i,j}$ denotes the j-th character in p_i, $c_{i,<j}$ denotes all the proceeding characters that has been built in p_i. We define $P_\theta(c_{i,j} | c_{i,<j}, p_{<i}, \mathbf{x}) = \alpha^{\mathrm{A}} + \alpha^{\mathrm{C}}$, where α^{A} and α^{C} are scores of the attention and the classifier, respectively. The prediction of dependency arc and type is taken as joint learning at each time.

3.6 Testing

In order to guarantee a valid dependency tree at test time, each character should have only one head except the dummy root "$". We introduce a list of "available" characters in the decoder. At each decoding step, the model selects a child for the current head, and removes the child from the list of available characters to make sure that it cannot be selected as a child of other head. For implementation of the Reduce action, the head character is temporarily recovered considering it has been removed before.

4 Experiments

4.1 Setup

We evaluate our model on the Penn Chinese Treebank (CTB version 5.0). At first, we process the data to obtain intra-word dependency structure. For one word, each character except the last, is annotated as the child of the last character with relation type tag "IN". The tag of the last character of the word is annotated as "OUT" suffixed with the POS tag of the word. The statistical information of the processed data is shown in Table 1.

Table 1 The statistics of the processed dataset

Dataset	Sentence	Word (Inter-word Arc)	Intra-word Arc
Training	16k	494k	311k
Development	352	6.8k	4.7k
Test	348	8.0k	5.7k

We use standard metrics 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 $\mathrm{F} = 2\mathrm{PR}/(\mathrm{P} + \mathrm{R})$ [18]. 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 segmented correctly.

4.2 Hyper-parameter

We use word2vec to pre-train the character vectors on the Gigaword, and the dimension of character vectors is 500 dimensions. The dimension of LSTM hidden layer is 512. The layers' number in encoder is 3, and in decoder is 1. We set batch size is 32. Adam is used in the optimization algorithm, with the initial learning rate is 0.001, and dropout value is 0.33. The learning rate is annealed by multiplying a fixed decay rate 0.75 when parsing performance stops increasing on development sets. To reduce the effects of "gradient exploding", we use gradient clipping of 5.0 [19].

4.3 Results

Higher-order Information. In order to choose the best model, we first compare the accuracies with and without grandfather and sibling in decoder. We further investigate the way of weight setting, one case with the frozen weights $\gamma_h = \gamma_g = \gamma_s = 1$, the other case with the unfrozen weights by adjusted with model training. The results are shown in the Table 2, where the "base" means a model without grandfather and sibling. From the table, we found that the best results on three tasks are those obtained by the model "base" using an unfrozen weight for $\gamma_h = 0.08$. The low value of the weight γ_h implies that previously generated information is more important in prediction. And the accuracies of the model using higher-order information are not improved like Ma et al. [9]. This best model will be used for subsequent comparisons with other works.

Table 2 Accuracy with/without higher-order information

model	weight	SEG	POS	DEP
base	Freeze	95.73	91.40	79.81
+grand	$(\gamma_h = \gamma_g = \gamma_s = 1)$	95.66	91.44	79.33
+sibling		95.64	91.06	78.97
+grand+sibling		95.63	91.21	79.22
base	$\gamma_h = 0.08$	**95.97**	**91.72**	**80.25**
+grand	$\gamma_h = 0.11\,; \gamma_g = 0.01$	95.88	91.56	79.94
+sibling	$\gamma_h = 0.08\,; \gamma_s = 0.04$	95.67	91.39	79.78
+grand+sibling	$\gamma_h = 0.10\,; \gamma_g = 0.01; \gamma_s = 0.05$	95.66	91.44	79.66

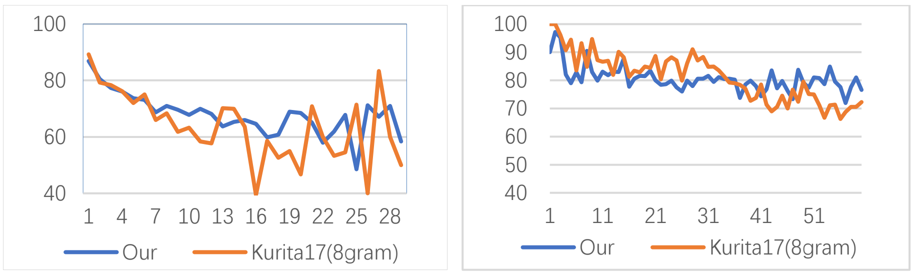
Main Comparison. We conduct comparison of our model with other joint parsing models and show the results in Table 3. The comparison models include Hatori12[4] and Zhang14 (character-level dependency parser based on feature engineering)[5], Kurita17 (joint dependency model combined with feature engineering for feature extraction and neural network for decision), Kurita17 (4-gram), Kurita17 (8-gram) [6]and Li19 (pure neural joint model)[7].

Table 3 Comparison results with the existing joint dependency parsing models

Model	SEG	POS	DEP
Hatori12	97.75	94.33	81.56
Zhang14	97.67	94.28	**81.65**
Kurita17	**98.24**	**94.49**	80.15
Kurita17(4-gram)	**97.72**	93.12	79.03
Kurita17(8-gram)	97.70	**93.37**	79.38
Li18	96.64	92.88	79.44
Ours(base)	95.92	91.65	**80.25**

The table is divided into two parts. The models in the top part are based on feature engineering and those in the bottom part are based entirely on neural network. From the table, we see that our model outperformed the existing neural joint model in dependency parsing tasks by 0.81%, and is closer to the feature engineering methods using a large number of artificially designed feature templates. At the same time, we also found that the accuracies of word segmentation and POS tagging are lower than other neural models. The observed results show that the proposed top-down character-level dependency architecture can significantly improve the accuracy of dependency parsing despite the lower accuracies on word segmentation and POS tagging.

Length. We compare our model with the reproduced Kurita17(8-gram) model according to dependency length and sentence length respectively. The results are shown in Fig. 3. From the figure, we can see that our model has better accuracy on long-distance dependency and long sentences, which shows that our model effectively extract global information and identify and construct long-distance dependency. At the same time, we find that our dependency accuracy on short sentences is lower. We investigate the results and analyze the reason is the low accuracy of word segmentation.



(a) Dependency length (b) Sentence length

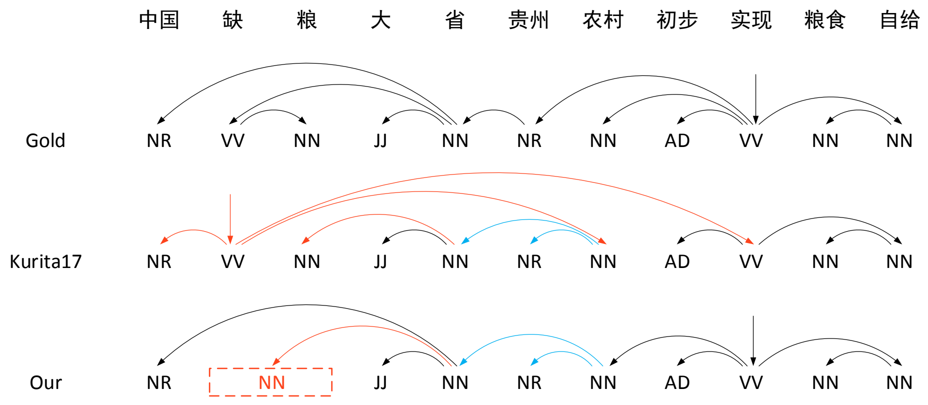
Fig. 3. The influence of length on dependency parsing

Case analysis. In this part, we try to analyze the difference between the top-down model(our model) and the classical transition-based model(Kurita17) through an instance. As shown in fig. 4, we take the sentence "中国缺粮大省贵州农村初步实现粮食自给 (In Guizhou, China's major grain shortage province, rural areas have initially achieved self-sufficiency in grain.)" as an example. The following are the results of word segmentation, POS tagging and dependency parsing respectively got from the gold standard, Kurita17 and our model.

Firstly, these orange lines for Kurita17 represent the prediction errors. The errors are due to that Kurita17 analyzes "中国缺粮大省贵州农村 (Guizhou rural areas, China's major grain shortage province)" as a sub-sentence, and "初步实现粮食自给 (have initially achieved self-sufficiency in grain)" as the other sub-sentence. Specifically, Kurita17 predicts the relationship between "缺 (lack)" and "农村 (rural areas)" as a right arc “缺→农村”, but "缺粮 (grain shortage)" should be an attributive modifier of "省 (province)" (as seen in Gold). In the prediction between "缺 (lack)" and "实现 (achieve)", because the first verb is usually the root of a sentence when there are multiple verbs in a sentence in CTB corpus.

Secondly, our model does not have the above errors due to the top-down model framework that enables it to correctly identify the root. However, since verbs rarely modify nouns, our model predicts "缺粮 (grain shortage)" as a word with POS tag “NN”. This may be the reason why our model performs poorly in word segmentation task.

Finally, the blue arcs are the dependency arcs that Kurita17 and our model both predicted wrong. This mistake is mainly caused by the omission of the sentence, because the complete expression of the original sentence should be “在中国缺粮大省贵州，农村初步实现粮食自给”, while the continuous nouns “贵州 (Guizhou)” and “农村 (rural areas)” in the sentence are usually generated a left arc “贵州←农村”, so neither model can correctly predict.



中国 缺 粮 大 省 贵州 农村 初步 实现 粮食 自给
Gold
NR VV NN JJ NN NR NN AD VV NN NN
Kurita17
NR VV NN JJ NN NR NN AD VV NN NN
Our
NR NN JJ NN NR NN AD VV NN NN

Fig. 4 Forecast Results of Different models for “中国缺粮大省贵州农村初步实现粮食自给”

5 Related Work

Hatori et al. [4] proposed a character-level dependency architecture for the first time, which combines word segmentation, part-of-speech 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. Zhang et al [5] annotated the internal structure of words for the Penn Chinese Treebank (CTB5), and regarded the word segmentation task as dependency parsing within characters to jointly process with dependency parsing. And they achieved the best accuracy in dependency parsing task at present.

Kurita et al. [6] applied neural network to the character-level dependency parsing for the first time. They used feature templates, pre-trained character and word vectors to extract distributed features, which were input into a multi-layer perceptron(MLP) to make decisions on parsing actions. They achieved the best results at present in word segmentation and part-of-speech tagging tasks. At the same time, they also used BiLSTM to encode the N-gram information to realize automatic feature extraction, with accuracies close to the best results of feature engineering on three tasks. Li et al.

[7] used the tagged part of speech and intra-word dependency of Chinese characters to process character-level dependency parsing with Stack-LSTM [20]. Their model outperformed the models of Kurita17(8-gram) on dependency parsing task.

The above described models are based on classical transition-based framework, which has a restriction of left-to-right. Ma et al. [9] proposed a novel transition-based model performing parsing in an incremental, top-down, depth-first fashion. The model recursively searches for child nodes from the root downward, making full use of global information, and breaking through the limitation of the classical transition-based models that can only consider the top two words of the stack and the first word of the buffer for decision.

Inspired by the work of Ma et al. [9], we propose a top-down character-level Chinese dependency parsing architecture to joint three tasks together, which overcomes the drawbacks of the traditional transition-based dependency parsers, and solves the problem of global information utilization and error propagation in long-distance dependence.

6 Conclusion

This paper proposes a top-down character-level Chinese dependency framework. The word segmentation task is realized by constructing the intra-word relation type, and the POS tagging task is realized by introducing POS tags into the inter-word relation type. In this way, a top-down joint model of Chinese word segmentation, POS tagging and dependency parsing is realized. Since our model has a global view for the input sentence, the information of the whole sentence and all previously generated dependency arcs are available for action decisions, and all characters of the sentence are considered as candidate children, without the left-to-right restriction. Our evaluation results on the CTB5 dataset outperform the existing neural network models in the dependency parsing. The model is also evaluated on higher-order information, sentence and dependency length, and children order. In the future, we will improve the joint model under the top-down dependency parsing framework, and make our model more robust for word segmentation and part-of-speech tagging, so as to further improve the accuracy of dependency parsing.

Acknowledgments. The authors are supported by the National Nature Science Foundation of China (Nos. 61876198, 61370130 and 61473294), the Fundamental Research Funds for the Central Universities (2015JBM033), and the International Science and Technology Cooperation Program of China (No. K11F100010).

References

1. Ma X, Liu Z, and Hovy E. Unsupervised ranking model for entity coreference resolution. In Proceedings of NAACL-2016. San Diego, California, USA. (2016)

2. Bastings J, Titov I, Aziz W, Marcheggiani D, and Simaan K. Graph convolutional encoders for syntax-aware neural machine translation. In Proceedings of EMNLP-2017. Copenhagen, Denmark, pages 1957–1967. (2017)
3. Peng N, Poon H, Quirk C, Toutanova K, and Yih W. Cross-sentence n-ary relation extraction with graph lstms. Transactions of the Association for Computational Linguistics 5:101–115. (2017)
4. Hatori J., Matsuzaki T., Miyao Y., Tsujii J.: Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese. In: Proceedings of ACL, pp.1045-1053. Association for Computational, Jeju (2012).
5. Zhang M., Zhang Y., Che W., Liu T.: Character-level Chinese dependency parsing. In: Proceedings of ACL, pp.1326-1336. Association for Computational, Baltimore (2012).
6. Kurita S., Kawahara D., Kurohashi S.: Neural joint model for transition-based Chinese syntactic analysis. In: Proceedings of ACL, pp.1204-1214. Association for Computational, Vancouver, (2018).
7. Li H., Zhang Z., Ju Y., Zhao H.: Neural character-level dependency parsing for Chinese. In: Proceedings of 32nd AAAI Conference on Artificial Intelligence (2018).
8. Mcdonald, R., & Nivre, J.: Analyzing and integrating dependency parsers. Computational Linguistics, 37(1), 197-230 (2011).
9. Ma X., Hu Z., Liu J., Peng N., Neubig G., Hovy E.: Stack-pointer networks for dependency parsing. In: Proceedings of ACL, pp.1403-1414. Association for Computational, Melbourne (2018).
10. Vinyals O., Fortunato M., Jaitly N.: Pointer Networks. Computer Science, (2015).
11. Fernández-González D., Gómez-Rodríguez C.: Left-to-Right Dependency Parsing with Pointer Networks. arXiv preprint arXiv:1903.08445 (2019).
12. Zheng X., Chen H., Xu T.: Deep learning for Chinese word segmentation and POS tagging. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp: 647-657 (2014).
13. Shao Y., Hardmeier C., Tiedemann J., Nivre J.: Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF. arXiv preprint arXiv:1704.01314 (2017).
14. Hochreiter S., Schmidhuber J.: Long short-term memory. Neural computation 9(8), 1735-1780 (1997).
15. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., JonesL., Gomez A. N., Kaiser L., PolosukhinI.: Attention is all you need. In: Advances in neural information processing systems, pp: 5998-6008 (2017).
16. Luong M T., Pham H., Manning C. D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015).
17. Dozat T., Manning C D.: Deep biaffine attention for neural dependency parsing. arXiv preprint arXiv:1611.01734 (2016).
18. Jiang, W., Huang, L., Liu, Q., Yajuan Lü: A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging. In: Proceedings of ACL. Association for Computational Linguistics, Columbus (2008).
19. Pascanu R., Mikolov T., Bengio Y.: On the difficulty of training recurrent neural networks. International conference on machine learning, pp: 1310-1318 (2013).
20. Dyer C., Ballesteros M., Ling W., Matthews A., Smith N. A.: Transition-based dependency parsing with stack long short-term memory. arXiv preprint arXiv:1505.08075 (2015).