

Neural Character-level Dependency Parsing for Chinese

Haonan Li^{1,2}, Zhisong Zhang^{1,2}, Yuqi Ju^{1,2}, Hai Zhao^{1,2,*}

¹Department of Computer Science and Engineering, Shanghai Jiao Tong University

²Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering

Shanghai Jiao Tong University, Shanghai, 200240, China

nathan_l@163.com, {zsz2011, tongkong}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

This paper presents a truly full character-level neural dependency parser together with a newly released character-level dependency treebank for Chinese, which has suffered a lot from the dilemma of defining word or not to model character interactions. Integrating full character-level dependencies with character embedding and human annotated character-level part-of-speech and dependency labels for the first time, we show an extra performance enhancement from the evaluation on Chinese Penn Treebank and SJTU (Shanghai Jiao Tong University) Chinese Character Dependency Treebank and the potential of better understanding deeper structure of Chinese sentences.

Introduction

Chinese language processing suffers from an obvious writing inconvenience: there is no clear separator between Chinese words while it is written in consecutive character sequence. People have to do a key character-level preprocessing before word-level parsing, i.e., word segmentation. Since (Zhao 2009) pointed out that Chinese parsing also receives the consequence of vague word definition, a series of work have considered character-level parsing (Zhao, Kit, and Song 2009; Li and Zhou 2012; Zhang et al. 2014).

Character-level dependency parsing, which was proposed as an alternative to word-level dependency parsing, has two benefits: 1) using character-level trees circumvents the issue that no universal standard exists for Chinese word segmentation. 2) in-depth structure inside word offers additional information for deeper level processing and better understanding of the whole sentence.

For the first benefit, it demonstrates in many aspects. On one hand, linguistic views about Chinese word standard diverge. Since the first SIGHAN Bakeoff shared task for Chinese word segmentation (Sproat and Emerson 2003), a lot

[†]Corresponding author. This paper was partially supported by National Natural Science Foundation of China (No. 61170114, No. 61672343 and No. 61733011), National Key Research and Development Program of China (No. 2017YFB0304100), Major Basic Research Program of Shanghai Science and Technology Committee (No. 15JC1400103), Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04), Key Project of National Society Science Foundation of China (No. 15-ZDA041). Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

of Chinese word segmentation standards have been proposed. Until Bakeoff-4 (Jin and Chen 2008), there were seven kinds of word segmentation conventions. On the other hand, whether a Chinese character sequence should be divided and the segmentation granularity are both blurred.


- (a) 中西医 / 结合
Chinese and Western medicine / integration
- (b) 中 / 西医
Chinese / Western medicine
- (c)  中 / 西 / 医
Chinese / Western / medicine

Figure 1: Example of segmentation choice.

Figure 1 illustrates a word segmentation case. Figure 1(a) shows that three-character segment 中西医 is as being a noun phrase in a sentence. Figure 1(b) gives an intuitive segmentation for the last two characters which are indeed a true word that means *Western medicine*. Figure 1(c) gives another segmentation (still unsatisfactory and problematic) in which each character is a single-character word. However, neither of the above segmentations are semantically proper, as the character sequence 中西医 actually means *Chinese medicine and Western medicine* and either the first character (meaning: *Chinese*) or the second (meaning: *Western*) is a modifier of the third one (meaning: *medicine*) just as shown as arcs in Figure 1(c). The above example shows that word segmentation decisions are not so easy in Chinese. In addition, all these problematic segmentations must confuse later syntactic parsing as word segmentation has a root position for all processing pipeline. Though in recent years, Chinese word segmentation as a sequence learning task has been made a success including our previous work (Zhao et al. 2010; Zhao 2011; Cai and Zhao 2016; Cai et al. 2017), the linguistic difficulties keep existing.

For the second benefit, as (Zhao 2009; Zhang et al. 2013) raised, a lot of Chinese words have internal structures which

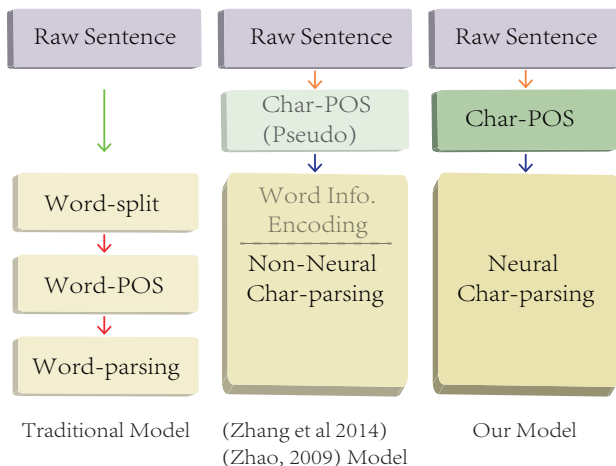


Figure 2: Traditional parsing pipeline and previous and our character dependency parsing models².

have been ignored ever since before. A Chinese word is defined by all the characters inside that subtly interact with each other from both syntax and semantics. For example, 窗(Window) and 帘(Screen) together mean 窗帘(Curtain) which indeed means *the Screen on the Window*. Thus the empirical results in (Zhang et al. 2014) indicate that including character-based information brings better performance for dependency parsing.

Exploring character dependency parsing with neural model, this paper shows three-fold contributions as follows:

(1) For the first time, Chinese character part-of-speech (POS) is taken into careful linguistics-motivated consideration. In fact, a Chinese character might represent quite different meanings in different words. For instance, in Chinese word 开花(bloom), the character 花 means *flowers*, while in the word 花费(expenditure), 花 refers to *spend*. Character with quite different multiple senses indicates that specific syntactic category (POS) should be assigned, which was never formally considered in computational linguistics as conventional Chinese processing starts from word and previous work on character dependency parsing also only defined trivial character-level POS (Zhang et al. 2014). The proposed parser will explore effectively integrating such kind of character POS and character dependency label.

(2) We provide a unified character-level dependency parsing framework with a purpose of giving a full structure decomposition for Chinese sentence once for all rather than enhancing word segmentation from the parsing angle as in (Zhao 2009; Li and Zhou 2012; Zhang et al. 2013; 2014). In fact, nearly all parsing models are still based on traditional multiple-stage workflow, in which each step takes input from the output of the previous step with inevitable substantial error accumulation. Our unified treatment streamlines this pipeline for more effective error propagation control. Figure 2 compares Chinese sentence processing pipelines between traditional parsing models and character-level parsing models.

(3) There have been a few work on character-based neural model (Zhang, Zhao, and Lecun 2015), but the term ‘character’ in these papers refers to English letters which play a similar role as strokes in Chinese³. We take characters embedding as the smallest units for Chinese dependency parsing. To our best knowledge, this is the first neural Chinese dependency parser at character level.

Dependency Parsing

Background

Dependency parsing aims to predict a dependency graph $G = (V, A)$ for the input sentence (Nivre and McDonald 2008). The input sentence is divided into segments in advance, and each segment corresponds to a node of the vertex set V . A represents the set of directed edges, which connect two nodes in the graph and illustrates the dependency relation of two nodes. Further, a relation label could be attached to the edge as the dependency type. In this way, dependency graph G eventually forms a tree. Usually, there are two constraints on the graph: acyclic, single-head (each node must have one and only one head node), and finally the predicted structure will be a well-formed dependency tree.

Conventionally, Chinese dependency parsing takes words as the input nodes, which needs a previous step of word segmentation. Our treatment does not need this step and formalizes it into a unified framework starting from characters.

Parsing Models

There are two typical models for dependency parsing: graph-based models which explore the entire space of dependency trees but with limited features, and transition-based models which exploit rich features but can only explore partial spaces. The graph-based dependency parser first scores for all the sub-tree structures, then it searches for a valid dependency graph space and produces the most likely one (with the highest score). Though dynamic programming can solve the problem with exact solution, the global search has relatively high time complexity. Since character-level parsing confronts longer sentences due to word decomposed into more characters, the high computational cost will be a challenge. Therefore, we adapt transition models, which search for an approximate solution with less cost.

In detail, the parse trees in transition methods are built by sequential shift-reduce like transitions. These transitions modify the current parsing state, and with a series of valid transitions, they could incrementally build the partial dependency structures until the final well-formed tree. However, it is hard to find the best transition sequence exactly for a huge search space, thus approximate strategies like greedy search or beam search are adopted in practice. These methods usually take linear time to the length of the sentence n , which is

²Word tagging information is copied to all internal characters for helping build a full character-level dependency tree.

³Our work is not a simple extension to Chinese. To be exact, Chinese strokes make up Chinese characters and Chinese characters make up Chinese word, while English letters constitute the English word directly.

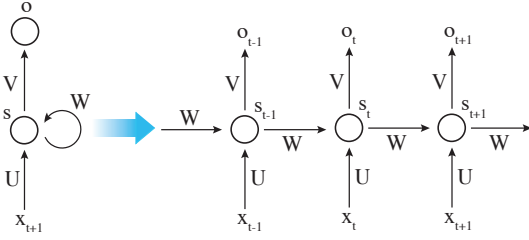


Figure 3: Recurrent neural network through time steps.

suitable for character-level parsing where n could be much larger than word-level parsing.

Neural Parser

For both graph-based and transition-based models, neural methods have been proved effective.

Among various neural models, recurrent neural network (RNN) has been extensively explored in the natural language processing field, for it naturally encodes the time sequential property in languages. Figure 3 illustrates the time-step expansion of RNN, where each input x stands for the tokens in the language sequence. Transition-based models also solve the problem with the left-to-right sequential decisions and take one token at a time, where RNN is quite suitable for memorizing the history and making the decisions.

Long Short-Term Memory (LSTM) is a special type of RNN, which can effectively solve the exploding gradient problem (Hochreiter and Schmidhuber 1997; Graves, Fernandez, and Schmidhuber 2005). In many problems, LSTM has achieved considerable success. (Dyer et al. 2015) proposed a model utilizing a stack LSTM in transition-based parsing, which encodes the parsing history and partial tree with stack LSTM. This model is especially suitable for character-level parsing, since with the expanding of words into characters, more long-range dependencies have to be recognized and LSTM is right well known for remembering histories. Thus we utilize this model as the off-the-shelf tool in our Chinese character-level parsing.

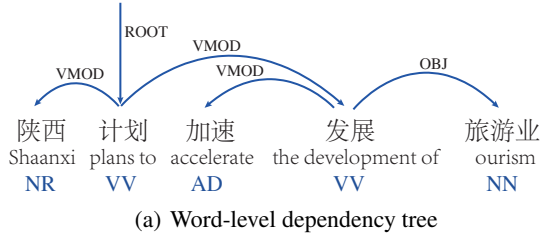
Character-level Dependency Parsing

Traditionally, Chinese dependency parsing takes words as the input tokens and predicts dependencies upon them, which is the so-called word-level parsing. However, this calls for a previous step of word segmentation, which could bring inevitable error inputs to the parsing step. Moreover, as discussed above, ignoring the internal structure of Chinese words indeed gives away important information. Therefore, character-level dependency parsing will take advantages over word-level parsing from the both points.

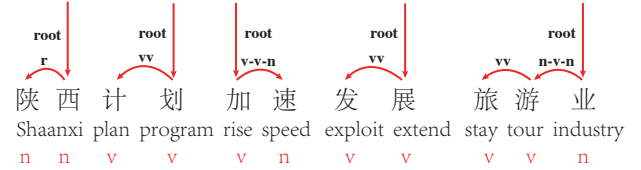
Character-level Dependency Treebank

To train the character-level parsing model, we bundle two treebanks to construct the required full character-level dependency treebank.

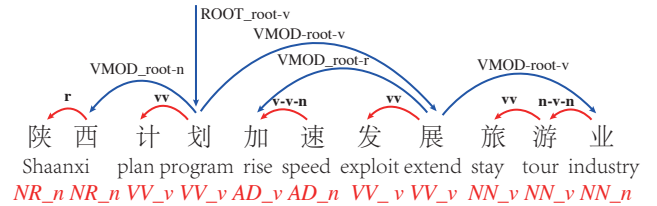
At the word-level, as in traditional methods, dependency treebank is converted from Chinese Penn Treebank (CTB),



(a) Word-level dependency tree



(b) Word-internal dependency trees



(c) Character-level dependency tree

Figure 4: Character-level dependency tree. We use single-character POS tags such as n, v, p , etc. to represent character POS and dependency labels, as two identical character POS tags to be the same resulted POS tag, such as $v+v \rightarrow v$, we use a simplified writing vv instead of $v-v-v$.

as shown in Figure 4(a). At the character-level, the dependencies will be from SJTU (Shanghai Jiao Tong University) Chinese Character Dependency Treebank (SCDT⁴), which includes character-level dependencies for all words of CTB.

SCDT directly gives the dependency structures inside the words (hereafter we refer to word-internal dependencies), and each word is annotated with a dependency tree on its characters, as shown in Figure 4(b). SCDT also defines a set of POS tags for characters, as in Table 1. For the character-level POS tagging, an iterative annotation strategy is utilized, adopting head rules and dynamic tagging method (with consideration of the context). For the dependency labels, SCDT basically follows Collins' convention, i.e., combinations of constituent labels (Collins 2010) as shown in Figure 4(c)⁵.

⁴It was annotated according to a pre-determined linguistically-motivated guideline on detailed character-level dependencies which is also released along with the Treebank. The treebank is available at <http://bcmi.sjtu.edu.cn/~zebraform/scdt.html>

⁵As most words (more than 93%) in Chinese text are one or two character long. We see many root-character labels from the figure.

Annotation	POS tag	Example
p	Pronoun	这(this), 那(that), 我(I), 一(one)
n	Noun	门(door), 体(body), 名(name)
i	Number and other characters	1, 2, ...
v	Verb	写(write), 跑(run)
a	Adjective	红(red), 苦(difficult)
d	Adverbial	很(very), 最(most)
f	Functional character	的(of), 们(-es), 在(at)

Table 1: The SCDT POS tag set.

Bundling Character-level and Word-level Dependencies

With both character- and word-level dependencies, the rest task is to bundle them into a character-level dependency tree.

Bundling Unlabeled Dependencies Considering character dependencies inside word, each word has its own root character, which may be used to stand for the word in the word-level treebank, then character dependencies and word dependencies can be naturally connected without any ambiguity. As shown in Figure 4, the final character-level tree will be obtained by combining all the trees: 4(a) is its word-level dependency tree, 4(b) gives all related character-level dependency trees for every words, and 4(c) shows character-level dependency tree for the whole sentence.

Bundling Character POS tags and Dependency labels Characters that belong to a word may inherit the corresponding word-level POS and dependency label from word-level treebank. Meanwhile, each character has its own character-level tag or label from character-level treebank. A full character dependency tree is to decide which detailed POS tag or dependency label, character-level, word-level or both, should be assigned to each character. Four tagging strategies will be introduced as follows by taking the situations of POS tags as example.

For each character, it has a word-level POS tag (this belongs to the word it stays) from CTB, and a character POS tag from SCDT. Note both of the POS tags are from manual annotations. (Zhang et al. 2014) also explored a character POS tagging strategy, trivially according to character position and word POS tag annotated by CTB⁶, while in this work character tags or labels for use are manually annotated and thus capable of representing the true syntactical role of the character inside the word.

To determine character-level POS tags from word-level and character-level annotations, four strategies are considered as in Figure 5.

- Word-level POS only. (Figure 5(a))
- Character-level POS only. (Figure 5(b))
- Combining word-level and character-level for all characters. (Figure 5(c))

⁶For example, a Chinese word has POS tag NN, then the first character is labeled as NN-b, and all the rest character is labeled as NN-i, where -b indicates the beginning character position and -i indicates others. We will denote this tagging strategy as WORD+CHAR-POSIT.

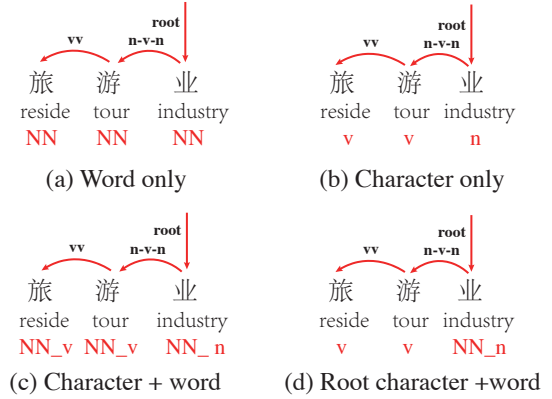


Figure 5: Four strategies to tag character POS.

- Combining word-level and character-level only for root characters. (Figure 5(d))

For the character-level dependency labels, the same four strategies will be considered.

With the enhanced character tagging strategies, the resulted character dependency tree expresses both lexical and syntactic information. As for word-level dependency parsing, the learning pipeline includes identification of words and recognition of each word’s attributes and dependencies. As to character analysis, we only need to train the dependencies of a character sequence. The error propagation in different processing hierarchies will be greatly reduced which is supposed to lead to a better learning effect.

Character Embedding

For character-level dependency parsing, when using neural models, we will need to represent characters instead of words. For representation in neural models, word embeddings are the common choice.

The proposed model uses character as the basic element for neural learning, so we naturally use character embeddings instead of word embeddings to feed neural models. Character embeddings and word embeddings are the same in principle for training and testing. In fact, character embeddings can be more conveniently trained than word embeddings for Chinese by trivially segmenting each character as single-character word.

Experiments

Settings

Data We use Chinese Penn Treebank 5.1 (CTB5) for evaluation. Dataset splitting follows (Zhang and Clark 2008). For preparing the character-level treebank, the word-level dependencies are obtained using Penn2Malt converter⁷; Then, combining the character dependencies from SCDT, the final character-level dependencies are obtained.

Model We compare traditional model and neural model for the character-level and word-level dependency parsing, respectively. MaltParser (Nivre, Hall, and Nilsson 2006) with default settings for the traditional model is exploited, while for the neural one, the LSTM parser in (Dyer et al. 2015) are utilized. The hyper-parameters are set as follows: 50 for character embedding⁸, 10 for POS embedding, 20 for action embedding and 3 for LSTM hidden layers.

Evaluation Although at present, there is no general standards to evaluate full character-level dependency parsing, we can still follow the practice of word-level parsing and take UAS/LAS (unlabeled/labeled attachment scores) on character-level tokens as the metrics. However, as the parsing element has been shifted from word to character, there will not be a meaningful comparison between word UAS and character UAS. This inconvenience makes us do necessary restorations from character-level dependency parsing results such as restoring words, which will be described later.

Character-level dependency parsing covers all levels of language processing within a Chinese sentence. Our model simplifies the pipeline into two steps, character POS tagging, and character dependency parsing, while traditional processing has to handle word segmentation, POS tagging for word, and word-level dependency parsing as shown in Figure 2. With different processing hierarchies, we also provide complete matches (CM) as one metric for the related evaluation.

Character-level Tagging Strategies

Before parsing, the character POS tags are learned and annotated with a Conditional Random Fields (CRFs) tagger using the same features and settings of (Chen, Zhang, and Sun 2008), in which all word n -gram features are transformed into corresponding character features⁹.

To determine the most useful character tagging strategy, character parsing performance comparison is given in Table 2, in which the following observations are obtained.

Firstly, with effective character POS tagging strategy, the parsing performance greatly increases, namely, 20% improvement over the one without any character POS tag fea-

⁷<http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html>

⁸No pretrained character or word embeddings are used for a fair comparison and verifying the effectiveness of the model alone. As our character-level parser only uses character embedding, no word embeddings are involved.

⁹In fact, (Chen, Zhang, and Sun 2008) used a maximum entropy Markov model as sequence labeling tool for word-level POS tagging instead of CRFs. However, both maximum entropy model and CRFs share the similar feature representations.

Tagging Strategies	#POS	#Label	UAS	LAS
NO TAGS	0	1	68.92	-
WORD-ONLY (Li 2011)	31	12	79.49	71.80
WORD+CHAR-POSIT (Zhang et al. 2014)	57	24	88.61	80.67
CHAR-ONLY	8	67	71.87	67.07
CHAR+WORD	134	549	87.24	80.31
ROOT-CHAR+WORD	121	141	89.51	87.45

Table 2: Results (on Dev set) of different tagging strategies (# stands for the number of POS tags or dependency labels).

tures, indicating that character-level syntactic annotation is indeed helpful for the parsing task.

Secondly, though WORD-ONLY outperforms CHAR-ONLY, using either character-level or word-level tags alone does not perform well. Note that WORD-ONLY strategy is right the one in (Li 2011) and (Zhang et al. 2013). WORD+CHAR-POSIT used by (Zhang et al. 2014) further shows that even integrating the least character position information, it is beneficial to the parser.

Finally, effective integration of two levels of tags boosts the performance most. For CHAR+WORD strategy, it is more straightforward but also brings too many tags or labels and thus will slow down the parsing and make the learning more difficultly, while the best performing ROOT-CHAR+WORD with less POS tags and much less dependency labels is immune from the drawbacks, which will be therefore adopted for the rest experiments.

Character-level Evaluation

Character-level parsing results on test set are given in Table 3, in which we see that using the same greedy transition parsing the neural model gives better accuracies. The reason might be that since characters instead of words are treated as the basic tokens, the task asks model to remember much more token history during parsing and the LSTM is the right one for the purpose. Besides, the character-level parsing seriously depends on the character-level POS tagging, as we see that the parser with golden character POS greatly outperforms those without any character POS features or moderately-predicted POS tags.

For reference, we also list highest UAS in (Zhang et al. 2014) who used a transition parser on their own character-level treebank that does not provide character-level POS and dependency label annotation. Note that character-level POS in (Zhang et al. 2014) just encodes word boundary extracted from segmentation annotation, while we provide human annotated rich character-level POS, using the same traditional model, our parser still gives better performance for the better character-level POS source.

Word-level Evaluation

To evaluate word-level performance, we perform two types of comparisons. One is about previous related work by restoring word-level parsing trees, the other is a pipeline processing for exploring the impact of error propagation.

	char POS	Parser		CM
		UAS	LAS	
(Zhang et al., 2014)	–	82.07	–	–
Char MaltParser	100.00	83.72	82.19	22.04
Char LSTM Parser	100.00	90.00	87.91	34.36
	90.12	82.53	80.47	21.54
	–	80.09	–	19.28

Table 3: Character-level evaluation.

To restore word-level information, we adopt a heuristic method to transform our character-level parsing results to word-level¹⁰. Our strategy is simple: finding all characters which are rooted by a character whose character-level POS tag indicates Root (according to the annotation of ROOT-CHAR+WORD). The process starts from each root character node by checking if its tag includes word-level POS part. For example, if a character POS tag is *NN_n*, then the corresponding character must be a root character as *NN* is known as a word-level POS tag. Note that word-level POS tag is determined for the soon later determined word. Once the root character is identified all its descendant nodes (with character-level only POS tags) can be collected and together with this character itself a word will be composed. All these edge subtrees collapsing into words with corresponding POS tags, a word-level dependency tree as in Figure 6 can be finally built.

We compare the accuracies for the restored trees with previous work in Table 5, in which character-level constituent parsing (Li 2011) is also given for reference. The restored word-level information subject to the same golden standard shows that our model only gives competitive performance on word segmentation, word-level POS tagging and word-level dependency parsing¹¹, no matter our parser presents much more informative character-level dependency structures below word than either of previous work.

Moreover, we evaluate our character-level parser in a pipeline way. BaseSeg and BasePoS (Zhao, Huang, and Li 2006) are utilized to respectively perform word segmentation and POS tagging tasks for further word-level dependency parsing. The comparisons are shown in Table 4. We list two types of results, the first is the process based on golden input of previous step, which means the input of each step is completely correct and thus immune from error propagation, the second is the process taking inputs from previous steps¹². All pipelines are supposed to provide a full pars-

¹⁰As character-level dependency tree covers word-level information, we could always extract or restore the word-level dependencies from it, although our character-level dependency model does not really rely on the concept of word.

¹¹According to the word restoring procedure, both performance of the reported word segmentation and word-level POS tagging here are mostly determined by our character POS tagger, which suggests that a better Character POS tagger may show further better results here. We leave this line to the future work.

¹²In this pipeline case, F-score is used to calculate UAS/LAS. In case a word is wrongly segmented, all later related part, word POS tag, unlabeled/labeled dependencies will be considered wrong, that

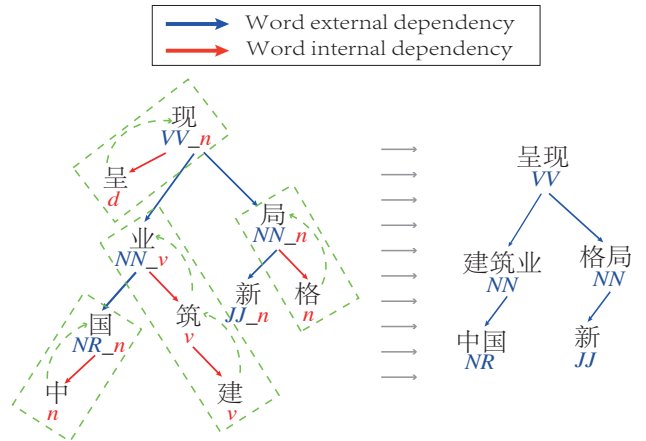


Figure 6: Word-level dependencies restore using ROOT-CHAR+WORD tagging

ing tree at last, which makes comparable word-level parsing results (including CM). We see that our character-level parser outperforms both word-level parsers with the golden character-level POS, especially, there is a CM comparison between 34.36 and 26.86. Still, we see that character-level parsing seriously relies on the character POS. For a better character parser, the corresponding character POS tagger should be well built.

Related Work and Discussion

Traditional dependency parsing models usually use linear models and sparse features, while recently, neural network models have gained popularity for its ability of automatic feature engineering, which show more effective in either graph-based (Zhang, Zhao, and Qin 2016) and transition-based parsing models (Chen and Manning 2014; Weiss et al. 2015). All these methods concern only word-level parsing, while this paper focuses on character-level parsing. For Chinese character-level parsing, the length of the sentence could be much longer than conventional word-level parsing, this makes parsing harder because the transition decisions need contexts with larger length. LSTM parsers such as (Dyer et al. 2015) are thus more suitable for the task with its recurrent structure modeling the entire sentence.

For Chinese parsing, there generally are two solutions to tackle the word definition ambiguity and too many process hierarchies over a sentence. The first is to perform a joint learning task for all levels of processing from word segmentation, POS tagging to parsing (Hatori et al. 2012; Qian and Liu 2012). We regard this as a kind of computation-motivated solution. The second solution is quite different, which seeks help from the linguistic root of Chinese language and includes this work as well. This type of related work was pioneered by (Zhao 2009). Character-level parsing for sentence is a linguistics-motivated rewriting scheme

means a complete error over the word.

	e.p.	Char- POS	Word seg.	Word- POS	Parser				CM
					UAS		LAS		
					word	char	word	char	
Word MaltParser	no	–	95.24	92.46	77.55	–	76.04	–	–
	yes	–	95.24	89.74	74.31	–	72.78	–	24.35
Word LSTM Parser	no	–	95.24	92.46	84.40	–	83.14	–	–
	yes	–	95.24	89.74	80.66	–	79.39	–	26.86
Char LSTM Parser	no	90.12	–	–	84.54	90.00	83.16	87.91	34.36
	yes	90.12	–	–	79.44	82.53	77.35	80.47	21.54

Table 4: Pipelined analysis. (e.p. indicates the processing is based on predicted results of previous steps.)

System	Word seg.	Word- POS	Parse	
			Phrase	Dep.
(Li 2011)	97.3	93.5	79.7	–
(Zhang et al. 2014)	97.84	94.62	–	82.14
Our model	96.64	92.88	–	79.44

Table 5: Word-level evaluation (with restoration), Dep. indicates UAS for dependency parsing.

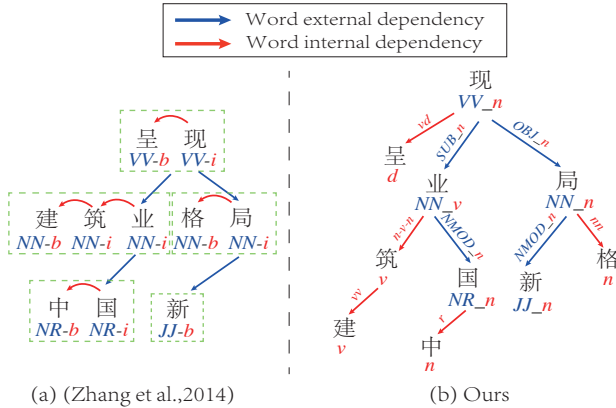


Figure 7: Character-level dependency trees.

for Chinese hierarchical processing, it alleviates the wordhood definition dilemma and proposes more natural and simple computational framework at the same time. (Zhao 2009) proposed that internal dependencies inside word can be helpful for Chinese word segmentation, and he also suggested integrating character-level dependencies into word-level dependency tree for a unified parsing scheme. The suggestion has been soon implemented in (Zhao, Kit, and Song 2009), which is the first work considering that a full character dependency parsing can give better CM than traditional word segmentation starting pipeline process. Later, this research has received further studied in two branches. One is the work of (Li and Zhou 2012) and (Zhang et al. 2013) who considered using a constituent style of internal structure for word and focused on constituent parsing, the other is (Zhang et al. 2014) who again considered character dependency parsing in a unified way as (Zhao 2009) suggested and what (Zhao, Kit, and Song 2009) implemented in a different character treebank. (Zhang et al.

2014) also gave important insight and empirical results to show that a linguistics-motivated character-level model essentially outperformed the computation-motivated joint learning models for traditional workflow from word segmentation to word-level parsing such as (Hatori et al. 2012; Qian and Liu 2012).

This paper is still on the line of the character dependency parsing. For the first time, character embedding is adopted for such a task. Meanwhile, an open character dependency treebank with character POS and dependency labels, SCDT, is publicly released and firstly made a full use. Actually what this work differs from all previous character-level structure exploring is more than the mode improvement. As shown in Figure 7, character-level information, POS tags or dependency labels in previous related work, is only copying from word-level treebank, which actually makes their models the ones only with word boundaries nominally encoded as dependencies (or phrase structures) for the benefits of joint learning in a unified computational framework¹³, while this work instead considers a *linguistically-new* task with newly introduced character-level POS tags and dependency labels and demonstrates that a truly character initializing parsing model from both linguistic and computational motivations can perform better with offering more informative structures for Chinese sentence analysis.

Conclusion

This paper presents an open character-level dependency treebank, SCDT, which first provides rich character-level POS and dependency annotations, and the first neural character-level dependency parser for Chinese.

Our empirical comparison shows that character-level POS tags and dependency labels play an important role for parsing performance, which were neglected in previous work or trivially treated due to incomplete character dependency treebank. Neural character dependency parsing is also shown to be more effective than none-NN parser in terms of main parsing metrics. The comparisons demonstrate that the proposed model gives promising performance aiming at a better understanding of Chinese sentences.

¹³Previous character-level parsing models more focused on improving word segmentation through joint learning rather than better parsing or exploring deeper structures of Chinese sentences.

References

- Cai, D., and Zhao, H. 2016. Neural word segmentation learning for chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 409–420.
- Cai, D.; Zhao, H.; Zhang, Z.; Xin, Y.; Wu, Y.; and Huang, F. 2017. Fast and accurate neural word segmentation for chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 608–615.
- Chen, D., and Manning, C. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 740–750.
- Chen, A.; Zhang, Y.; and Sun, G. 2008. A two-stage approach to Chinese part-of-speech tagging. In *the Sixth SIGHAN Workshop on Chinese Language Processing (SIGHAN)*, 82–85.
- Collins, M. 2010. Head-driven statistical models for natural language parsing. *Computational Linguistics* 29(4):589–637.
- Dyer, C.; Ballesteros, M.; Ling, W.; Matthews, A.; and Smith, N. A. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 334–343.
- Graves, A.; Fernández, S.; and Schmidhuber, J. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal MODELS and Their Applications (ICANN)*, 799–804.
- Hatori, J.; Matsuzaki, T.; Miyao, Y.; and Tsujii, J. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in Chinese. In *Proceedings of the 50rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 1045–1053.
- Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.
- Jin, G., and Chen, X. 2008. The fourth international Chinese language processing bakeoff: Chinese word segmentation, named entity recognition and Chinese pos tagging. In *the Sixth SIGHAN Workshop on Chinese Language Processing (SIGHAN)*, 69–81.
- Li, Z., and Zhou, G. 2012. Unified dependency parsing of Chinese morphological and syntactic structures. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 1445–1454.
- Li, Z. 2011. Parsing the internal structure of words: a new paradigm for Chinese word segmentation. In *Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, 1405–1414.
- Nivre, J., and McDonald, R. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*, 950–958.
- Nivre, J.; Hall, J.; and Nilsson, J. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceeding of Language Resources and Evaluation Conference (LREC)* 2216–2219.
- Qian, X., and Liu, Y. 2012. Joint Chinese word segmentation, pos tagging and parsing. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 501–511.
- Sproat, R., and Emerson, T. 2003. The first international Chinese word segmentation bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing (SIGHAN)*, 133–143.
- Weiss, D.; Alberti, C.; Collins, M.; and Petrov, S. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 323–333.
- Zhang, Y., and Clark, S. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 562–571.
- Zhang, M.; Zhang, Y.; Che, W.; and Liu, T. 2013. Chinese parsing exploiting characters. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 125–134.
- Zhang, M.; Zhang, Y.; Che, W.; and Liu, T. 2014. Character-level Chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 1326–1336.
- Zhang, X.; Zhao, J.; and Lecun, Y. 2015. Character-level convolutional networks for text classification. In *Neural Information Processing Systems (NIPS)*, 649–657.
- Zhang, Z.; Zhao, H.; and Qin, L. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 1382–1392.
- Zhao, H.; Huang, C. N.; Li, M.; and Lu, B. L. 2010. A unified character-based tagging framework for chinese word segmentation. *Acm Transactions on Asian Language Information Processing* 9(2):1–32.
- Zhao, H.; Huang, C. N.; and Li, M. 2006. An improved Chinese word segmentation system with conditional random field. In *the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN)*, 162–165.
- Zhao, H.; Kit, C.; and Song, Y. 2009. Character dependency tree based lexical and syntactic all-in-one parsing for chinese. In *The 10th Chinese National Conference on Computational Linguistics (CNCCL-2009)*, 82–88.
- Zhao, H. 2009. Character-level dependencies in Chinese: Usefulness and learning. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL)*, 879–887.
- Zhao, H. 2011. Integrating unsupervised and supervised word segmentation: The role of goodness measures. *Information Sciences* 181(1):163–183.